SAP TECHED '05

MAKING ABAP PROGRAMS UNICODE ENABLED

SPC250

EXERCISES / SOLUTIONS

DR. CHRISTIAN HANSEN, NETWEAVER DEVELOPMENT TOOLS INTERNATIONALIZATION, SAP AG

Contents

The programs you have to inspect are ZTECHED_UNICODE_EXERCISE_n_XX where n are the numbers 1 to 10 and XX indicates the two digit group number. Please don't touch the original exercise patterns TECHED_UNICODE_EXERCISE_n.
Possible solutions are found in the programs TECHED_UNICODE_SOLUTION_n.
(You may find the exercises and solutions in any SAP system based on release SAP_BASIS 6.10 or higher).

Part I : Using UCCHECK to remove static Unicode syntax errors

- Logon to the non-Unicode system
- Inspect the programs ZTECHED_UNICODE_EXERCISE_1/2/3/4/5_XX using the transaction UCCHECK.
- Remove all static Unicode errors and set the Unicode attribute.

Hints for the exercise:

- ✓ In the object selection uncheck the option "Exclude $* Packages"

Part II: Find critical places with UCCHECK

- Logon to the non-Unicode system
- Run the program ZTECHED_UNICODE_EXERCISE_6_XX, check for static errors with UCCHECK and set the Unicode attribute.
- Try to rerun the program.
- Analyze the program with the UCCHECK option "statically not analyzable places".
- Remove warnings by properly typing parameters and variables.
- Remove Unicode runtime errors.

Hints for the exercise:

- ✓ Use "#EC * to hide warnings from places that cannot be removed by proper typing.
- ✓ Use generic types for the subroutine F00.

Part III: Using SCOV to screen runtime tests

- Logon to the Unicode system
- Look in the Coverage Analyzer (transaction SCOV) for the list of processing blocks of program ZTECHED_UNICODE_EXERCISE_7_XX
- Execute the program several times and look in the Coverage Analyzer for the coverage of the different processing blocks.
- Find the errors in form F100 and F11.

Hints for the exercise:

- ✓ If you want to cover all subroutines, you should look at the program coding with the debugger. It is just a simple tree structure.
- ✓ If you don't want to spend too much time, try the following number combinations
  - o 0/0; 0/1; 1/0/1/0; 1/0/1/1; 1/1
- ✓ Finally, try the following paths:
  - o 1/0/42 (Unicode error, visible only in a Unicode system)
  - o 1/13 (Data dependent error, not Unicode specific)

Part IV: Look at the ABAP list layout

- Logon to the Unicode system
- Execute the programs ZTECHED_UNICODE_EXERCISE_8/9/10_XX. Find and understand the pitfalls of the shown list programming techniques.
- Remove the errors.

Technical information:

The exercises are based on the following Releases/Patchlevels

- Minimum: basis 6.20 Unicode system Support Package >=21, Kernel Patch level >= 401
- Minimum: Basis 6.20 non Unicode system Support Package >=21, Kernel Patch level >= 401, Konfiguration as MDMP System (RSCPINST) with at least German, Japanese, Korean and English Locale installed
- Minimum: SAPGUI 6.40
- Users were created with transaction BC_TOOLS_USER
- The coverage analyzer is switched on activate data collection at SCOV → Administration → "On/Off, Status" → Switch Coverage Analyzer On/Off, Button "On" . Before the "Data Collection: Bckgrd Server" has been maintained at SCOV → Administration → Settings
- In order to show and experiment with multilangual data you may create a table with the fields COLOR,/SPRAS/NAME with the types CHAR 1 / LANG 1 / CHAR 6 and fill the table with report RSCPCOLORS . This may be done in both a non Unicode MDMP and a Unicode system. To display the content use SE16 → F6 → ALV Grid → Enter → F7 → F8.
- Exercises in the customer name space "ZTECHED_UNICODE_EXERCISE_n_XX" are prepared with the report TECHED_UNICODE_EXERCISE_COPY

Listings: Exercises

Exercise 1

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_EXERCISE_1                                   *
*&---------------------------------------------------------------------*
REPORT teched_unicode_exercise_1.

*** Exercise 1 (easy): Distinction between byte and character length
*** Hint: Use one of the additions
***        ... IN BYTE MODE or
***        ... IN CHARACTER MODE
***       for the DESCRIBE ... LENGTH statements.


*** before Unicode enabling
DATA:
  p1(4) TYPE p VALUE 1234,
  p2(8) TYPE p DECIMALS 3 VALUE '3.141'.


PERFORM test1 USING 'abcdef'.


PERFORM test2 USING p1.
PERFORM test2 USING p2.

*---------------------------------------------------------------------*
*  FORM test1
*---------------------------------------------------------------------*
FORM test1 USING text TYPE c.
  DATA: len TYPE i,
        off TYPE i.

  DESCRIBE FIELD text LENGTH len.

  WHILE len > 0.
    WRITE: / text+off(1).
    len = len - 1.
    off = off + 1.
  ENDWHILE.
ENDFORM.                                                    "test1


*---------------------------------------------------------------------*
*  FORM test2
*---------------------------------------------------------------------*
FORM test2 USING val TYPE p.
  DATA: len TYPE i,
        decs TYPE i.

  DESCRIBE FIELD val LENGTH len
                     DECIMALS decs.
  ULINE.
  WRITE: / 'Value    =', val.
  WRITE: / 'Length   =', len.
  WRITE: / 'Decimals =', decs.
ENDFORM.                                                    "test2
```

Exercise 2

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_EXERCISE_2                                   *
*&---------------------------------------------------------------------*
REPORT teched_unicode_exercise_2.

*** Exercise 2: String processing
*** Hint: Use attributes MINCHAR and CR_LF of class
***       CL_ABAP_CHAR_UTILITIES

*** before Unicode enabling

CONSTANTS: hex0(1) TYPE x VALUE '00',
           crlf(2) TYPE x VALUE '0D0A'.

DATA: BEGIN OF line1,
        text1(10)  TYPE c VALUE 'system:   ',
        mark1(1)   TYPE x VALUE hex0,
        text2(10)  TYPE c VALUE 'user:     ',
        mark2(1)   TYPE x VALUE hex0,
        text3(10)  TYPE c VALUE 'client:   ',
        mark3(1)   TYPE x VALUE hex0,
        space(100) TYPE c,
      END OF line1,
      line2(133) TYPE c.

REPLACE: hex0 WITH sy-sysid INTO line1,
         hex0 WITH sy-uname INTO line1,
         hex0 WITH sy-mandt INTO line1.

CONDENSE line1.

CONCATENATE line1 crlf INTO line1.

line2 = line1.

WRITE: / line2.
```

Exercise 3

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_EXERCISE_3                                   *
*&---------------------------------------------------------------------*
REPORT teched_unicode_exercise_3.

*** Exercise 3: Accessing structures with offset or length
*** Hint: Introduce an include with a group name in order to
***       avoid accessing the structure with offset/length.

*** before Unicode enabling

DATA:
  BEGIN OF struc1,
    f0     TYPE i,
    f1(10) TYPE c,
    f2(10) TYPE c,
    f3(10) TYPE c,
    f4(10) TYPE c,
    f5(10) TYPE c,
    f6     TYPE p,
  END OF struc1,

  cf(50) TYPE c,

  BEGIN OF struc2,
    f0 TYPE i VALUE 42,
    f1 TYPE i VALUE 12,
    f2 TYPE i VALUE 34,
    f3 TYPE i VALUE 56,
    f4 TYPE i VALUE 78,
    f5 TYPE i VALUE 90,
  END OF struc2.

MOVE-CORRESPONDING struc2 TO struc1.

cf = struc1+4(50).

CONDENSE cf.
WRITE: / cf.
```

Exercise 4

```
REPORT teched_unicode_exercise_4 .

*** Exercise 4: Move / Data container
*** Hint: Replace container-contents by an X-string
***       and use
***          EXPORT ... TO DATA BUFFER ...
***          IMPORT ... FROM DATA BUFFER ...
***       for storing and reading the contents of the container.



*** before Unicode enabling

TYPES:
  BEGIN OF person,
    firstname(20) TYPE c,
    lastname(20)  TYPE c,
    age           TYPE i,
  END OF person,

  BEGIN OF address,
    street(20)  TYPE c,
    city(30)    TYPE c,
    code(6)     TYPE c,
    country(20) TYPE c,
  END OF address,

  BEGIN OF container,
    tag(1)        TYPE c,
    contents(100) TYPE c,
  END OF container.



DATA:
  container_tab TYPE STANDARD TABLE OF container.

PERFORM fill_table  CHANGING container_tab.

PERFORM write_table USING container_tab.


*---------------------------------------------------------------------*
*  FORM fill_table
*---------------------------------------------------------------------*
FORM fill_table CHANGING data_tab LIKE container_tab.

  DATA:
    pers TYPE person,
    addr TYPE address,
    entry TYPE container.

  pers-firstname = 'Max'.
  pers-lastname  = 'Meier'.
  pers-age       = 33.
  entry-tag      = 'P'.
  entry-contents = pers.
  APPEND entry TO data_tab.
```

```
  addr-street    = 'Hauptstrasse 10'.
  addr-city      = 'Wien'.
  addr-code      = '12345'.
  addr-country   = 'Austria'.
  entry-tag      = 'A'.
  entry-contents = addr.
  APPEND entry TO data_tab.

  addr-street    = 'Brucknerstrasse 11'.
  addr-city      = 'Linz'.
  addr-code      = '4321'.
  addr-country   = 'Austria'.
  entry-tag      = 'A'.
  entry-contents = addr.
  APPEND entry TO container_tab.


  pers-firstname = 'Fred'.
  pers-lastname  = 'Smith'.
  pers-age       = 66.
  entry-tag      = 'P'.
  entry-contents = pers.
  APPEND entry TO data_tab.
ENDFORM.                    "fill_table


*---------------------------------------------------------------------*
*  FORM write_table
*---------------------------------------------------------------------*
FORM write_table USING data_tab LIKE container_tab.
  DATA: container_line TYPE container,
        pers TYPE person,
        addr TYPE address.

  LOOP AT data_tab INTO container_line.
    CASE container_line-tag.
      WHEN 'P'.
        pers = container_line-contents.
        WRITE: / 'First name: ', pers-firstname.
        WRITE: / 'Last name:  ', pers-lastname.
        WRITE: / 'Age:        ', (3) pers-age.
        ULINE.

      WHEN 'A'.
        addr = container_line-contents.
        WRITE: / 'Street:     ', addr-street.
        WRITE: / 'City:       ', addr-city.
        WRITE: / 'Postal Code:', addr-code.
        WRITE: / 'Country:    ', addr-country.
        ULINE.
    ENDCASE.
  ENDLOOP.
ENDFORM.                    "print_table
```

Exercise 5

```
REPORT teched_unicode_exercise_5 .

*** Exercise 5 (difficult): Dynamic database operations
*** Hint: Using the following ABAP statements for
***       dynamic programming:
***         CREATE DATA ... TYPE (tname)
***         DESCRIBE FIELD ... TYPE ... COMPONENTS n
***         ASSIGN COMPONENT index OF STRUCTURE ... TO ...
***       you can rewrite the program in a clear and concise way
***       without any use of function 'DDIF_NAMETAB_GET' and
***       without any ASSIGN with offset/length to workarea of type C.


*** before Unicode enabling


*** This programs lists the contents of all field of a database table
*** up to the given number of rows.

PARAMETERS:
  tabname(30)       TYPE c DEFAULT 'SPFLI', "name of database table
  limit             TYPE i DEFAULT 10.      "maximal number of rows

DATA:
  alignment_dummy   TYPE f,
  container(1000)   TYPE c,
  dfies_tab         TYPE TABLE OF dfies.

FIELD-SYMBOLS: <dt> TYPE dfies,
               <f>  TYPE ANY.

* DDIC_NAMETAB_GET returns an internal table containing a desciption of
* the fields of a table. Offsets and length of each field is
* always counted in bytes.
CALL FUNCTION 'DDIF_NAMETAB_GET'
  EXPORTING
    tabname   = tabname
  TABLES
    dfies_tab = dfies_tab.

WRITE: / 'Table:', tabname.
SELECT * FROM (tabname) INTO container UP TO limit ROWS.
  WRITE: /(4) sy-dbcnt, '|'.
  LOOP AT dfies_tab ASSIGNING <dt>.
    ASSIGN container+<dt>-offset(<dt>-intlen)
           TO <f>
           TYPE <dt>-inttype.
    WRITE  <f>.
  ENDLOOP.
ENDSELECT.
```

Exercise 6

```
*&---------------------------------------------------------------------*
*& Report TECHED_UNICODE_EXERCISE_6                                    *
*&---------------------------------------------------------------------*
report teched_unicode_exercise_6.

data: g_step type i value 1.

perform start.
skip.
write :/ 'The program finished without runtime errors.'.


*---------------------------------------------------------------------*
*  FORM start
*---------------------------------------------------------------------*
form start.

  data: l_number type i,
        l_mod    type i,
        l_raw(5) type x value '4A4B4C4D4E',
        begin of l_struc,
           type(3) type c value 'ABC',
           value   type p value 1000,
        end of l_struc.

  write :/ 'This is Subroutine START'.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f0 using l_struc.
  else.
    perform f1 using l_raw.
  endif.

endform.                    "start

*---------------------------------------------------------------------*
*  FORM f0
*---------------------------------------------------------------------*
form f0 using p.

  data: l_number  type i,
        l_mod     type i,
        l_text    type string.

  write :/ 'This is Subroutine F0'.

  concatenate 'I got the structure: ' p into l_text.
  write :/ l_text.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_text.
  else.
    perform f01 using l_number.
  endif.

endform.                    "f0
```

```
*---------------------------------------------------------------------*
*  FORM f1
*---------------------------------------------------------------------*
form f1 using p.

  data: l_number     type i,
        l_mod        type i,
        l_hex_to_int type i.

  write :/ 'This is Subroutine F1'.

  l_hex_to_int = p.
  write :/ 'Hexadecimal: ', p.
  write :/ 'Decimal    : ', l_hex_to_int.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f10 using p.
  else.
    perform f11 using l_number.
  endif.

endform.                    "f1

*---------------------------------------------------------------------*
*  FORM f00
*---------------------------------------------------------------------*
form f00 using p.

  write :/ 'This is Subroutine F00'.

  write :/ p.
  p = 'Hello1'.
  p = 'Hello2'.
  p = 'Hello3'.
  p = 'Hello4'.
  p = 'Hello5'.
  p = 'Hello6'.

endform.                    "f00

*---------------------------------------------------------------------*
*  FORM f01
*---------------------------------------------------------------------*
form f01 using p.

  data: l_text1(30)  type c,
        l_text2(30) type c.

  write :/ 'This is Subroutine F01'.
  write p to l_text1.
  write :/ l_text1.

  clear l_text2 with 'A'.
  if p > 30. p = 30. endif.
  l_text1 = l_text2(p).
  write :/ l_text1.


endform.                    "f01
```

```
*---------------------------------------------------------------------*
*  FORM f10
*---------------------------------------------------------------------*
form f10 using p.

  data: l_number  type i,
        l_mod     type i,
        l_text(7) type c value 'A B C D'.


  write :/ 'This is Subroutine F10'.

  if p ca 'J'.
    write :/ 'J is in ', p.
  else.
    write :/ 'J is not in ', p.
  endif.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    check l_number = 42.
    perform f100 using l_number.
  else.
    perform f101 using l_text.
  endif.

endform.                    "f10

*---------------------------------------------------------------------*
*  FORM f11
*---------------------------------------------------------------------*
form f11 using p type i.

  data: l_text(10) type c value '0123456789'.

  write :/ 'This is Subroutine F11'.

  write :/ l_text(p).

endform.                    "f11


*---------------------------------------------------------------------*
*  FORM f100
*---------------------------------------------------------------------*
form f100 using p type i.

  data: l_len      type i,
        l_text(26) type c value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.

  write :/ 'This is Subroutine F100'.

  describe field l_text length l_len in byte mode.
  subtract 1 from l_len.
  write :/ l_text(l_len).

endform.                                                    "f100

*---------------------------------------------------------------------*
*  FORM f101
*---------------------------------------------------------------------*
form f101 using p.
```

```
  data: l_number type i,
        l_mod    type i,
        l_date   type sydatum value '20010926',
        l_num(5) type n       value '12345'.


  write :/ 'This is Subroutine F101'.

  write :/ 'Before CONDENSE NO-GAPS:', p.
  condense p no-gaps.
  write :/ 'After  CONDENSE NO-GAPS:', p.



  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_date.
  else.
    perform f00 using l_num.
  endif.
endform.                                                    "f101


*---------------------------------------------------------------------*
*  FORM enter_value
*---------------------------------------------------------------------*
form enter_value changing p_number p_mod.


  data: l_fields         type table of sval,
        l_fields_wa      type sval,
        l_rc             type c,
        l_stepsc(3)      type c,
        l_text(30)       type c.


  p_number = 0.
  p_mod    = 0.

  l_fields_wa-tabname   = 'SYST'.
  l_fields_wa-fieldname = 'SUBRC'.
  l_fields_wa-value     = ' '.
  clear l_fields.
  append l_fields_wa to l_fields.
  write g_step to l_stepsc.
  concatenate 'Step' l_stepsc ': PLease enter a number !' into
      l_text.
  g_step = g_step + 1.
  call function 'POPUP_GET_VALUES'
    exporting
      popup_title     = l_text
    importing
      returncode      = l_rc
    tables
      fields          = l_fields
    exceptions
      error_in_fields = 1
      others          = 2.

  if l_rc = 'A'.
    write :/ 'No number was entered. Bye !'.
    stop.
  endif.
```

```
  read table l_fields index 1 into l_fields_wa.
  p_number = l_fields_wa-value.

  write :/ 'You entered:', p_number.
  skip.
  write :/ sy-uline.

  p_mod = p_number mod 2.

endform.                    "enter_value
```

Exercise 7

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_EXERCISE_7                                   *
*&---------------------------------------------------------------------*

report teched_unicode_exercise_7.

types: begin of t_struc1,
           type(3) type c,
           value   type p,
        end of t_struc1.

data: g_step type i value 1.

perform start.
skip.
write :/ 'The program finished without runtime errors.'.


*---------------------------------------------------------------------*
*  FORM start
*---------------------------------------------------------------------*
form start.

  data: l_number type i,
        l_mod    type i,
        l_raw(5) type x value '4A4B4C4D4E',
        l_struc  type t_struc1.

  l_struc-type  = 'ABC'.
  l_struc-value = 1000.

  write :/ 'This is Subroutine START'.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f0 using l_struc.
  else.
    perform f1 using l_raw.
  endif.

endform.                    "start

*---------------------------------------------------------------------*
*  FORM f0
*---------------------------------------------------------------------*
form f0 using p type t_struc1.

  data: l_number    type i,
        l_mod       type i,
        l_text      type string,
        l_value(20) type c.

  write :/ 'This is Subroutine F0'.

  write p-value to l_value.
  concatenate 'I got a structure with the components: type '
              ' = ' p-type ' and value = ' l_value into l_text.
  write :/ l_text.
```

```
  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_text.
  else.
    perform f01 using l_number.
  endif.

endform.                    "f0

*---------------------------------------------------------------------*
*  FORM f1
*---------------------------------------------------------------------*
form f1 using p type x.

  data: l_number     type i,
        l_mod        type i,
        l_hex_to_int type i.

  write :/ 'This is Subroutine F1'.

  l_hex_to_int = p.
  write :/ 'Hexadecimal: ', p.
  write :/ 'Decimal    : ', l_hex_to_int.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f10 using p.
  else.
    perform f11 using l_number.
  endif.

endform.                    "f1

*---------------------------------------------------------------------*
*  FORM f00
*---------------------------------------------------------------------*
form f00 using p type clike.

  write :/ 'This is Subroutine F00'.

  write :/ p.
  p = 'Hello1'.
  p = 'Hello2'.
  p = 'Hello3'.
  p = 'Hello4'.
  p = 'Hello5'.
  p = 'Hello6'.

endform.                    "f00

*---------------------------------------------------------------------*
*  FORM f01
*---------------------------------------------------------------------*
form f01 using p type i.

  data: l_text1(30)  type c,
        l_text2(30) type c.

  write :/ 'This is Subroutine F01'.
  write p to l_text1.
  write :/ l_text1.
```

```
  clear l_text2 with 'A'.
  if p > 30. p = 30. endif.
  l_text1 = l_text2(p).                                    "#EC *
  write :/ l_text1.


endform.                    "f01

*---------------------------------------------------------------------*
*  FORM f10
*---------------------------------------------------------------------*
form f10 using p type x.

  data: l_number  type i,
        l_mod     type i,
        l_text(7) type c value 'A B C D',
        l_4a(1)   type x value '4A'.


  write :/ 'This is Subroutine F10'.

  if p byte-ca l_4a.
    write :/ 'Byte 4A is in ', p.
  else.
    write :/ 'Byte 4A is not in ', p.
  endif.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    check l_number = 42.
    perform f100 using l_number.
  else.
    perform f101 using l_text.
  endif.

endform.                    "f10

*---------------------------------------------------------------------*
*  FORM f11
*---------------------------------------------------------------------*
form f11 using p type i.

  data: l_text(10) type c value '0123456789'.

  write :/ 'This is Subroutine F11'.

  write :/ l_text(p).

endform.                    "f11


*---------------------------------------------------------------------*
*  FORM f100
*---------------------------------------------------------------------*
form f100 using p type i.

  data: l_len      type i,
        l_text(26) type c value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.

  write :/ 'This is Subroutine F100'.

  describe field l_text length l_len in byte mode.
  subtract 1 from l_len.
```

```
  write :/ l_text(l_len).

endform.                                                    "f100

*---------------------------------------------------------------------*
*  FORM f101
*---------------------------------------------------------------------*
form f101 using p type c.

  data: l_number type i,
        l_mod    type i,
        l_date   type sydatum value '20010926',
        l_num(5) type n       value '12345'.


  write :/ 'This is Subroutine F101'.

  write :/ 'Before CONDENSE NO-GAPS:', p.
  condense p no-gaps.
  write :/ 'After  CONDENSE NO-GAPS:', p.



  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_date.
  else.
    perform f00 using l_num.
  endif.
endform.                                                    "f101


*---------------------------------------------------------------------*
*  FORM enter_value
*---------------------------------------------------------------------*
form enter_value changing p_number type i
                          p_mod    type i.


  data: l_fields         type table of sval,
        l_fields_wa      type sval,
        l_rc             type c,
        l_stepsc(3)      type c,
        l_text(30)       type c.


  p_number = 0.
  p_mod    = 0.

  l_fields_wa-tabname   = 'SYST'.
  l_fields_wa-fieldname = 'SUBRC'.
  l_fields_wa-value     = ' '.
  clear l_fields.
  append l_fields_wa to l_fields.
  write g_step to l_stepsc.
  concatenate 'Step' l_stepsc ': PLease enter a number !' into
      l_text.
  g_step = g_step + 1.
  call function 'POPUP_GET_VALUES'
    exporting
      popup_title     = l_text
    importing
      returncode      = l_rc
```

```
    tables
      fields          = l_fields
    exceptions
      error_in_fields = 1
      others          = 2.

  if l_rc = 'A'.
    write :/ 'No number was entered. Bye !'.
    stop.
  endif.

  read table l_fields index 1 into l_fields_wa.
  p_number = l_fields_wa-value.

  write :/ 'You entered:', p_number.
  skip.
  write :/ sy-uline.

  p_mod = p_number mod 2.

endform.                    "enter_value
```

Exercise 8

```
*&---------------------------------------------------------------------
*& Report  TECHED_UNICODE_EXERCISE_8
*&---------------------------------------------------------------------report
teched_unicode_exercise_8.

start-of-selection.
  write :/ 'Please use a Korean Font to look at the list!'. "#EC NOTEXT

  perform:
   head using 'Example 1: SY-VLINE overwriting      ', example1,
   head using 'Example 2: self made right-justified ', example2,
   head using 'Example 3: display complete content  ', example3,
   head using 'Example 4: Container in memory layout', example4.

*&---------------------------------------------------------------------*
*&      Form  example1
*&---------------------------------------------------------------------*
form example1.

  new-line.
  write at 2 '11 22 33'.  "#EC NOTEXT
  write at 1  sy-vline.
  write at 4  sy-vline.
  write at 7  sy-vline.
  write at 10 sy-vline.

  new-line.
  write at 2 '일 이 삼'.
  write at 1  sy-vline.
  write at 4  sy-vline.
  write at 7  sy-vline.
  write at 10 sy-vline.
endform.                                                    "example1

*&---------------------------------------------------------------------*
*&      Form  example2
*&---------------------------------------------------------------------*
form example2.
  data: text(10) type c.
  text = 'hello'.
  write text to text right-justified.
  write :/ text.
  text =  '일이삼'.
  write text to text right-justified.
  write :/ text.
endform.                                                    "example2

*&---------------------------------------------------------------------*
*&      Form  example3
*&---------------------------------------------------------------------*
form example3.
  data: text(10) type c.
  text = 'OTTOS MOPS'.
  write:/  text.
  text = '가갸거겨고교구규그기'.
  write:/ text.
endform.                                                    "example3
```

```
*&---------------------------------------------------------------------*
*&      Form  example4
*&---------------------------------------------------------------------*
form example4.
  data: wa        type teched02_numbers,
        line(100) type c.
  select * from teched02_numbers into wa order by num lang.
    write wa-lang  to line(2).
    write sy-vline to line+2(1).
    write wa-name  to line+3(5).
    write sy-vline to line+8(1).
    write wa-num   to line+9(3) right-justified.
    write :/ line.
  endselect.
endform.                                                    "example4
*&---------------------------------------------------------------------*
*&      Form  head
*&---------------------------------------------------------------------*
form head  using text type clike.
  skip.
  write :/ sy-uline.
  write :/ text.
  write :/ sy-uline.
  skip.
endform.                    " head
```

Exercise 9

```
*&---------------------------------------------------------------------
*& Report  TECHED_UNICODE_EXERCISE_9
*&---------------------------------------------------------------------report
teched_unicode_exercise_9.

  selection-screen comment 1(80) head1.
  selection-screen skip.
  selection-screen comment 1(80) head2.
  selection-screen skip.
  selection-screen skip.
  selection-screen comment 1(80) lineal1.
  selection-screen skip.
  selection-screen comment 1(80) lineal2.
  selection-screen skip.
  selection-screen comment 1(80) text1.
  selection-screen skip.
  selection-screen comment 1(80) text2.
  selection-screen skip.
  selection-screen comment 1(80) text3.
  selection-screen skip.
  selection-screen comment 1(80) text4.
  selection-screen skip.
  selection-screen begin of line.
  selection-screen comment 1(17) sctext.
  selection-screen position 19.
  parameters: scrolcol type i default 15.
  selection-screen end of line.

  types t_line(100) type c.
  data: line type t_line,
        tab  type table of t_line,
        off  type i.

  head1 = 'Please use a Korean Font to look at the list!'. "#EC NOTEXT
  head2 = 'Example 4: self made scrolling'.                "#EC NOTEXT
  lineal1 = '....+....1....+....2....+....3'.
  lineal2 = '12345678901234567890123456789O'.
  text1 =   '이름1: xxxxxx 이름2: yyyyyy'.
  text2 =   '이름1: 남명희 이름2: 조홍우'.
  text3 = 'Name1: Schell Name2: Fuchs '.  "#EC NOTEXT
  text4 = 'Name1: Mayer  Name2: Novak '.  "#EC NOTEXT
  sctext = 'Scroll to column:'.           "#EC NOTEXT
  append: lineal1 to tab,
          lineal2 to tab,
          text1   to tab,
          text2   to tab,
          text3   to tab,
          text4   to tab.


start-of-selection.

  off = scrolcol - 1.
  loop at tab into line.
    write:/ line+off.
  endloop.
```

Exercise 10

```
*&---------------------------------------------------------------------
*& Report  TECHED_UNICODE_EXERCISE_10
*&---------------------------------------------------------------------
report teched_unicode_exercise_10.

initialization.

  data: text10_1 type string.
  import text to text10_1 from database teched02_unicode(ar) id 'text10_1'.

at line-selection.

  data: off     type i.
  off = sy-staco + sy-cucol - 3.
  sy-lisel+off(1) = 'x'.
  modify current line.

  read line 9 of current page.
  off = off + 1.
  write off to sy-lisel+10(10).
  modify line 9 of current page.

start-of-selection.

  data: text(100) type c.

  write :/ 'Please use a Korean Font to look at the list!'.  "#EC NOTEXT
  skip.

  write :/ 'Please replace all o with x by double clicking on it:'. "#EC NOTEXT
  skip.
  text = 'aa->o<-bbcc->o<-ddeeff->o<-gghhiijj'.                "#EC NOTEXT
  write /(50) text.
  text = text10_1.   "'가->o<-갸거->o<-겨고교->o<-구규그기'.
  write /(50) text.
  write / 'Column:'.                  "#EC NOTEXT
```

Solution - Exercise 1

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_1                                   *
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_1               .

*** Exercise 1: Distinction between byte and character length
*** after Unicode enabling

DATA:
  p1(4) TYPE p VALUE 1234,
  p2(8) TYPE p DECIMALS 3 VALUE '3.141'.


PERFORM test1 USING 'abcdef'.


PERFORM test2 USING p1.
PERFORM test2 USING p2.



*---------------------------------------------------------------------*
*  FORM test1
*---------------------------------------------------------------------*
FORM test1 USING text TYPE c.
  DATA: len TYPE i,
        off TYPE i.

*  describe field TEXT length LEN.     <------------- Unicode error!
  DESCRIBE FIELD text LENGTH len IN CHARACTER MODE.

  WHILE len > 0.
    WRITE: / text+off(1).
    len = len - 1.
    off = off + 1.
  ENDWHILE.
ENDFORM.                                                    "test1


*---------------------------------------------------------------------*
*  FORM test2
*---------------------------------------------------------------------*
FORM test2 USING val TYPE p.
  DATA: len TYPE i,
        decs TYPE i.

* DESCRIBE FIELD VAL LENGTH LEN          <------------- Unicode error!
*                    DECIMALS DECS
  DESCRIBE FIELD val LENGTH len IN BYTE MODE
                     DECIMALS decs
                     .
  ULINE.
  WRITE: / 'Value    =', val.
  WRITE: / 'Length   =', len.
  WRITE: / 'Decimals =', decs.
ENDFORM.                                                    "test2
```

Solution - Exercise 2

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_2                                   *
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_2               .

*** Exercise 2 - String processing
*** after Unicode enabling

CLASS cl_abap_char_utilities DEFINITION LOAD.

CONSTANTS:
  hex0(1) TYPE c VALUE cl_abap_char_utilities=>minchar,
  crlf(2) TYPE c VALUE cl_abap_char_utilities=>cr_lf.

DATA:
  BEGIN OF l_line1,
    text1(10)  TYPE c VALUE 'system:   ',
    mark1(1)   TYPE c VALUE hex0,
    text2(10)  TYPE c VALUE 'user:     ',
    mark2(1)   TYPE c VALUE hex0,
    text3(10)  TYPE c VALUE 'client:   ',
    mark3(1)   TYPE c VALUE hex0,
    space(100) TYPE c,
  END OF l_line1,
  l_line2(133) type c.

REPLACE: hex0 WITH sy-sysid INTO l_line1,
         hex0 WITH sy-uname INTO l_line1,
         hex0 WITH sy-mandt INTO l_line1.

CONDENSE l_line1.

CONCATENATE l_line1 crlf INTO l_line1.

l_line2 = l_line1.

WRITE :/ l_line2.
```

Solution - Exercise 3

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_3                                   *
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_3               .

*** Exercise 3: Accessing structures with offset or length
*** before Unicode enabling

types:
  begin of PART1,
      F1(10) type C,
      F2(10) type C,
      F3(10) type C,
      F4(10) type C,
      F5(10) type C,
  end of PART1.

data: begin of STRUC1,
        F0     type I.
include type PART1 as INCL1.
data:   F6     type P,
       end of STRUC1,

  CF(50) type C,

  begin of STRUC2,
    F0 type I value 42,
    F1 type I value 12,
    F2 type I value 34,
    F3 type I value 56,
    F4 type I value 78,
    F5 type I value 90,
  end of STRUC2.

move-corresponding STRUC2 to STRUC1.

* cf = struc1+4(50).  " <-------- Unicode error !!!
CF = STRUC1-INCL1.

condense CF.

write: / CF.
```

Solution - Exercise 4

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_4                                   *
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_4               .

*** Exercise 4: Move / Data container
*** after Unicode enabling

TYPES:
  BEGIN OF person,
    firstname(20) TYPE c,
    lastname(20)  TYPE c,
    age           TYPE i,
  END OF person,

  BEGIN OF address,
    street(20)  TYPE c,
    city(30)    TYPE c,
    code(6)     TYPE c,
    country(20) TYPE c,
  END OF address,

  BEGIN OF container,
    tag(1)        TYPE c,
*    contents(100) TYPE c,     <------------- changed!
    contents      TYPE xstring,
  END OF container.

DATA:
  container_tab TYPE STANDARD TABLE OF container.

PERFORM fill_table  CHANGING container_tab.

PERFORM write_table USING container_tab.


*---------------------------------------------------------------------*
*  FORM fill_table
*---------------------------------------------------------------------*
FORM fill_table CHANGING data_tab LIKE container_tab.

  DATA:
    pers_data TYPE person,
    addr_data TYPE address,
    entry     TYPE container.

  pers_data-firstname = 'Max'.
  pers_data-lastname  = 'Meier'.
  pers_data-age       = 33.
  entry-tag      = 'P'.
*  entry-contents = pers. <------------ Unicode error!
  EXPORT value = pers_data TO DATA BUFFER entry-contents.
  APPEND entry TO data_tab.

  addr_data-street    = 'Hauptstrasse 10'.
  addr_data-city      = 'Wien'.
  addr_data-code      = '12345'.
  addr_data-country   = 'Austria'.
  entry-tag      = 'A'.
```

```
*  entry-contents = addr.  <------------ Unicode error!
  EXPORT value = addr_data TO DATA BUFFER entry-contents.
  APPEND entry TO data_tab.

  addr_data-street    = 'Hauptstrasse 11'.
  addr_data-city      = 'Linz'.
  addr_data-code      = '4321'.
  addr_data-country   = 'Austria'.
  entry-tag      = 'A'.
*  entry-contents = addr.
  EXPORT value = addr_data TO DATA BUFFER entry-contents.
  APPEND entry TO container_tab.

  pers_data-firstname = 'Fred'.
  pers_data-lastname  = 'Smith'.
  pers_data-age       = 66.
  entry-tag      = 'P'.
*  entry-contents = pers.
  EXPORT value = pers_data TO DATA BUFFER entry-contents.
  APPEND entry TO data_tab.
ENDFORM.                    "fill_table


*---------------------------------------------------------------------*
*  FORM write_table
*---------------------------------------------------------------------*
FORM write_table USING data_tab LIKE container_tab.
  DATA: container_line TYPE container,
        pers TYPE person,
        addr TYPE address.

  LOOP AT data_tab INTO container_line.
    CASE container_line-tag.
      WHEN 'P'.
*        pers = container_line-contents.  <------------ Unicode error!
        IMPORT value = pers FROM DATA BUFFER container_line-contents.
        WRITE: / 'First name: ', pers-firstname.
        WRITE: / 'Last name:  ', pers-lastname.
        WRITE: / 'Age:        ', (3) pers-age.
        ULINE.

      WHEN 'A'.
*        addr = container_line-contents.
        IMPORT value = addr FROM DATA BUFFER container_line-contents.
        WRITE: / 'Street:     ', addr-street.
        WRITE: / 'City:       ', addr-city.
        WRITE: / 'Postal Code:', addr-code.
        WRITE: / 'Country:    ', addr-country.
        ULINE.
    ENDCASE.
  ENDLOOP.
ENDFORM.                    "write_table
```

Solution - Exercise 5

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_5                                   *
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_5               .

*** Exercise 5: Dynamic database operations

*** After unicode enabling.

PARAMETERS:
  tabname(30)       TYPE c DEFAULT 'SPFLI',
  limit             TYPE i DEFAULT 10.

*** deleted!
*DATA:
*  alignment_dummy   TYPE f,
*  container(1000)   TYPE c,
*  dfies_tab         TYPE TABLE OF dfies.
*
*FIELD-SYMBOLS: <dt> TYPE dfies.

FIELD-SYMBOLS:  <comp>  TYPE ANY.

DATA:          tabref      TYPE REF TO data,  "<---- inserted!
               compcnt     TYPE i,
               comptype(1) TYPE c.
FIELD-SYMBOLS: <tabline> TYPE ANY.          "<---- inserted!

*** deleted!
** DDIC_NAMETAB_GET returns an internal table containing a desciption of
** the fields a table. Offsets and length of each field is
** always given in bytes.
*CALL FUNCTION 'DDIF_NAMETAB_GET'
*  EXPORTING
*    tabname   = tabname
*  TABLES
*    dfies_tab = dfies_tab.

* create workarea of the given type
CREATE DATA tabref TYPE (tabname).
ASSIGN tabref->* TO <tabline>.

WRITE: / 'Table:', tabname.

*** deleted
*SELECT * FROM (tabname) INTO container UP TO limit ROWS.

*** replaced by select into dynamically created work area
SELECT * FROM (tabname) INTO <tabline> UP TO limit ROWS.


  WRITE: /(4) sy-dbcnt, '|'.

*** deleted: access to components with offset/length
*  LOOP AT dfies_tab ASSIGNING <dt>.
*    ASSIGN container+<dt>-offset(<dt>-intlen)
*           TO <comp>
*           TYPE <dt>-inttype.
*    WRITE  <comp>.
```

```
*  ENDLOOP.


*** new implementation: symbolic access to components
  DESCRIBE FIELD <tabline> TYPE comptype COMPONENTS compcnt.
  DO compcnt TIMES.
    ASSIGN COMPONENT sy-index OF STRUCTURE <tabline> TO <comp>.
    WRITE   <comp>.
  ENDDO.


ENDSELECT.
```

Alternative Solution - Exercise 5

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_5b
*&---------------------------------------------------------------------*
REPORT  teched_unicode_solution_5b               .

*** Exercise 5: Dynamic database operations

PARAMETERS:
  tabname(30)       TYPE c DEFAULT 'SPFLI',
  limit             TYPE i DEFAULT 10.

DATA:
  alignment_dummy   TYPE f,
  container(1000)   TYPE c,
  dfies_tab         TYPE TABLE OF dfies.
FIELD-SYMBOLS: <dt> TYPE dfies,
               <f>  TYPE any,
               <tabline> TYPE any,    "<---- new
               <x_container> type X.  "<---- new


* DDIC_NAMETAB_GET returns an internal table containing a desciption of
* the fields a table. Offsets and length of each field is
* always given in bytes.
CALL FUNCTION 'DDIF_NAMETAB_GET'
  EXPORTING
    tabname   = tabname
  TABLES
    dfies_tab = dfies_tab.

WRITE: / 'Table:', tabname.

assign container to <tabline> casting type (tabname). "<---- new
assign container to <x_container> casting.            "<---- new

*SELECT * FROM (tabname) INTO container UP TO limit ROWS. <----- error
SELECT * FROM (tabname) INTO <tabline> UP TO limit ROWS.
  WRITE: /(4) sy-dbcnt, '|'.
  LOOP AT dfies_tab ASSIGNING <dt>.
*    ASSIGN container+<dt>-offset(<dt>-intlen)
    ASSIGN <x_container>+<dt>-offset(<dt>-intlen)
           TO <f>
           TYPE <dt>-inttype.
    WRITE  <f>.
  ENDLOOP.
ENDSELECT.
```

Solution - Exercise 6

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_6                                   *
*&---------------------------------------------------------------------*
report teched_unicode_solution_6.

types: begin of t_struc1,
           type(3) type c,
           value   type p,
        end of t_struc1.

data: g_step type i value 1.

perform start.
skip.
write :/ 'The program finished without runtime errors.'.

*---------------------------------------------------------------------*
*  FORM start
*---------------------------------------------------------------------*
form start.

  data: l_number type i,
        l_mod    type i,
        l_raw(5) type x value '4A4B4C4D4E',
        l_struc  type t_struc1.

  l_struc-type  = 'ABC'.
  l_struc-value = 1000.

  write :/ 'This is Subroutine START'.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f0 using l_struc.
  else.
    perform f1 using l_raw.
  endif.

endform.                    "start

*---------------------------------------------------------------------*
*  FORM f0
*---------------------------------------------------------------------*
form f0 using p type t_struc1.

  data: l_number    type i,
        l_mod       type i,
        l_text      type string,
        l_value(20) type c.

  write :/ 'This is Subroutine F0'.

  write p-value to l_value.
  concatenate 'I got a structure with the components: type '
              ' = ' p-type ' and value = ' l_value into l_text.
  write :/ l_text.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
```

```
    perform f00 using l_text.
  else.
    perform f01 using l_number.
  endif.

endform.                    "f0

*---------------------------------------------------------------------*
*  FORM f1
*---------------------------------------------------------------------*
form f1 using p type x.

  data: l_number     type i,
        l_mod        type i,
        l_hex_to_int type i.

  write :/ 'This is Subroutine F1'.

  l_hex_to_int = p.
  write :/ 'Hexadecimal: ', p.
  write :/ 'Decimal    : ', l_hex_to_int.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f10 using p.
  else.
    perform f11 using l_number.
  endif.

endform.                    "f1

*---------------------------------------------------------------------*
*  FORM f00
*---------------------------------------------------------------------*
form f00 using p type clike.

  write :/ 'This is Subroutine F00'.

  write :/ p.
  p = 'Hello1'.
  p = 'Hello2'.
  p = 'Hello3'.
  p = 'Hello4'.
  p = 'Hello5'.
  p = 'Hello6'.

endform.                    "f00

*---------------------------------------------------------------------*
*  FORM f01
*---------------------------------------------------------------------*
form f01 using p type i.

  data: l_text1(30)  type c,
        l_text2(30) type c.

  write :/ 'This is Subroutine F01'.
  write p to l_text1.
  write :/ l_text1.

  clear l_text2 with 'A'.
  if p > 30. p = 30. endif.
  l_text1 = l_text2(p).                                  "#EC *
```

```
  write :/ l_text1.

endform.                    "f01

*---------------------------------------------------------------------*
*  FORM f10
*---------------------------------------------------------------------*
form f10 using p type x.

  data: l_number  type i,
        l_mod     type i,
        l_text(7) type c value 'A B C D',
        l_4a(1)   type x value '4A'.


  write :/ 'This is Subroutine F10'.

  if p byte-ca l_4a.
    write :/ 'Byte 4A is in ', p.
  else.
    write :/ 'Byte 4A is not in ', p.
  endif.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    check l_number = 42.
    perform f100 using l_number.
  else.
    perform f101 using l_text.
  endif.

endform.                    "f10

*---------------------------------------------------------------------*
*  FORM f11
*---------------------------------------------------------------------*
form f11 using p type i.

  data: l_text(10) type c value '0123456789'.

  write :/ 'This is Subroutine F11'.

  write :/ l_text(p).

endform.                    "f11


*---------------------------------------------------------------------*
*  FORM f100
*---------------------------------------------------------------------*
form f100 using p type i.

  data: l_len      type i,
        l_text(26) type c value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.

  write :/ 'This is Subroutine F100'.

  describe field l_text length l_len in byte mode.
  subtract 1 from l_len.
  write :/ l_text(l_len).

endform.                                                    "f100
```

```
*---------------------------------------------------------------------*
*  FORM f101
*---------------------------------------------------------------------*
form f101 using p type c.

  data: l_number type i,
        l_mod    type i,
        l_date   type sydatum value '20010926',
        l_num(5) type n       value '12345'.


  write :/ 'This is Subroutine F101'.

  write :/ 'Before CONDENSE NO-GAPS:', p.
  condense p no-gaps.
  write :/ 'After  CONDENSE NO-GAPS:', p.



  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_date.
  else.
    perform f00 using l_num.
  endif.
endform.                                                    "f101


*---------------------------------------------------------------------*
*  FORM enter_value
*---------------------------------------------------------------------*
form enter_value changing p_number type i
                          p_mod    type i.


  data: l_fields         type table of sval,
        l_fields_wa      type sval,
        l_rc             type c,
        l_stepsc(3)      type c,
        l_text(30)       type c.


  p_number = 0.
  p_mod    = 0.

  l_fields_wa-tabname   = 'SYST'.
  l_fields_wa-fieldname = 'SUBRC'.
  l_fields_wa-value     = ' '.
  clear l_fields.
  append l_fields_wa to l_fields.
  write g_step to l_stepsc.
  concatenate 'Step' l_stepsc ': PLease enter a number !' into
      l_text.
  g_step = g_step + 1.
  call function 'POPUP_GET_VALUES'
    exporting
      popup_title     = l_text
    importing
      returncode      = l_rc
    tables
      fields          = l_fields
    exceptions
      error_in_fields = 1
```

```
      others          = 2.

  if l_rc = 'A'.
    write :/ 'No number was entered. Bye !'.
    stop.
  endif.

  read table l_fields index 1 into l_fields_wa.
  p_number = l_fields_wa-value.

  write :/ 'You entered:', p_number.
  skip.
  write :/ sy-uline.

  p_mod = p_number mod 2.

endform.                    "enter_value
```

Solution - Exercise 7

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_7                                   *
*&---------------------------------------------------------------------*
report teched_unicode_solution_7.

types: begin of t_struc1,
           type(3) type c,
           value   type p,
        end of t_struc1.

data: g_step type i value 1.

perform start.
skip.
write :/ 'The program finished without runtime errors.'.


*---------------------------------------------------------------------*
*  FORM start
*---------------------------------------------------------------------*
form start.

  data: l_number type i,
        l_mod    type i,
        l_raw(5) type x value '4A4B4C4D4E',
        l_struc  type t_struc1.

  l_struc-type  = 'ABC'.
  l_struc-value = 1000.

  write :/ 'This is Subroutine START'.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f0 using l_struc.
  else.
    perform f1 using l_raw.
  endif.

endform.                    "start

*---------------------------------------------------------------------*
*  FORM f0
*---------------------------------------------------------------------*
form f0 using p type t_struc1.

  data: l_number    type i,
        l_mod       type i,
        l_text      type string,
        l_value(20) type c.

  write :/ 'This is Subroutine F0'.

  write p-value to l_value.
  concatenate 'I got a structure with the components: type '
              ' = ' p-type ' and value = ' l_value into l_text.
  write :/ l_text.

  perform enter_value changing l_number l_mod.
```

```
  if l_mod = 0.
    perform f00 using l_text.
  else.
    perform f01 using l_number.
  endif.

endform.                    "f0

*---------------------------------------------------------------------*
*  FORM f1
*---------------------------------------------------------------------*
form f1 using p type x.

  data: l_number     type i,
        l_mod        type i,
        l_hex_to_int type i.

  write :/ 'This is Subroutine F1'.

  l_hex_to_int = p.
  write :/ 'Hexadecimal: ', p.
  write :/ 'Decimal    : ', l_hex_to_int.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f10 using p.
  else.
    perform f11 using l_number.
  endif.

endform.                    "f1

*---------------------------------------------------------------------*
*  FORM f00
*---------------------------------------------------------------------*
form f00 using p type clike.

  write :/ 'This is Subroutine F00'.

  write :/ p.
  p = 'Hello1'.
  p = 'Hello2'.
  p = 'Hello3'.
  p = 'Hello4'.
  p = 'Hello5'.
  p = 'Hello6'.

endform.                    "f00

*---------------------------------------------------------------------*
*  FORM f01
*---------------------------------------------------------------------*
form f01 using p type i.

  data: l_text1(30)  type c,
        l_text2(30) type c.

  write :/ 'This is Subroutine F01'.
  write p to l_text1.
  write :/ l_text1.

  clear l_text2 with 'A'.
  if p > 30. p = 30. endif.
```

```
  l_text1 = l_text2(p).                                     "#EC *
  write :/ l_text1.


endform.                    "f01

*---------------------------------------------------------------------*
*  FORM f10
*---------------------------------------------------------------------*
form f10 using p type x.

  data: l_number  type i,
        l_mod     type i,
        l_text(7) type c value 'A B C D',
        l_4a(1)   type x value '4A'.


  write :/ 'This is Subroutine F10'.

  if p byte-ca l_4a.
    write :/ 'Byte 4A is in ', p.
  else.
    write :/ 'Byte 4A is not in ', p.
  endif.

  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    check l_number = 42.
    perform f100 using l_number.
  else.
    perform f101 using l_text.
  endif.

endform.                    "f10

*---------------------------------------------------------------------*
*  FORM f11
*---------------------------------------------------------------------*
form f11 using p type i.

  data: l_text(10) type c value '0123456789'.

  write :/ 'This is Subroutine F11'.

  if p > 10.
    p = 10.
    write :/ 'Number was restricted to 10 !'.
  endif.
  write :/ l_text(p).

endform.                    "f11


*---------------------------------------------------------------------*
*  FORM f100
*---------------------------------------------------------------------*
form f100 using p type i.

  data: l_len      type i,
        l_text(26) type c value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.

  write :/ 'This is Subroutine F100'.
```

```
  describe field l_text length l_len in character mode.
  subtract 1 from l_len.
  write :/ l_text(l_len).

endform.                                                    "f100

*---------------------------------------------------------------------*
*  FORM f101
*---------------------------------------------------------------------*
form f101 using p type c.

  data: l_number type i,
        l_mod    type i,
        l_date   type sydatum value '20010926',
        l_num(5) type n       value '12345'.


  write :/ 'This is Subroutine F101'.

  write :/ 'Before CONDENSE NO-GAPS:', p.
  condense p no-gaps.
  write :/ 'After  CONDENSE NO-GAPS:', p.



  perform enter_value changing l_number l_mod.
  if l_mod = 0.
    perform f00 using l_date.
  else.
    perform f00 using l_num.
  endif.
endform.                                                    "f101


*---------------------------------------------------------------------*
*  FORM enter_value
*---------------------------------------------------------------------*
form enter_value changing p_number type i
                          p_mod    type i.


  data: l_fields         type table of sval,
        l_fields_wa      type sval,
        l_rc             type c,
        l_stepsc(3)      type c,
        l_text(30)       type c.


  p_number = 0.
  p_mod    = 0.

  l_fields_wa-tabname   = 'SYST'.
  l_fields_wa-fieldname = 'SUBRC'.
  l_fields_wa-value     = ' '.
  clear l_fields.
  append l_fields_wa to l_fields.
  write g_step to l_stepsc.
  concatenate 'Step' l_stepsc ': PLease enter a number !' into
      l_text.
  g_step = g_step + 1.
  call function 'POPUP_GET_VALUES'
    exporting
      popup_title     = l_text
```

```
    importing
      returncode      = l_rc
    tables
      fields          = l_fields
    exceptions
      error_in_fields = 1
      others          = 2.

  if l_rc = 'A'.
    write :/ 'No number was entered. Bye !'.
    stop.
  endif.

  read table l_fields index 1 into l_fields_wa.
  p_number = l_fields_wa-value.

  write :/ 'You entered:', p_number.
  skip.
  write :/ sy-uline.

  p_mod = p_number mod 2.

endform.                    "enter_value
```

Solution - Exercise 8

```
*&---------------------------------------------------------------------
*& Report  TECHED_UNICODE_SOLUTION_8
*&---------------------------------------------------------------------report
teched_unicode_solution_8.

class cl_abap_list_utilities definition load.

start-of-selection.

  write :/ 'Please use a Korean Font to look at the list!'. "#EC NOTEXT

  perform:
   head using 'Solution 1: SY-VLINE overwriting',        solution1,
   head using 'Solution 2: self made right-justified',   solution2,
   head using 'Solution 3: display complete content',    solution3,
   head using 'Solution 4: Container in display layout', solution4.

*&---------------------------------------------------------------------*
*&      Form  solution1
*&---------------------------------------------------------------------*
form solution1.

  new-line.
  write at 1 sy-vline.
  write at 2 '11'.                                        "#EC NOTEXT
  write at 4 sy-vline.
  write at 5 '22'.                                        "#EC NOTEXT
  write at 7  sy-vline.
  write at 8 '33'.                                        "#EC NOTEXT
  write at 10 sy-vline.

  new-line.
  write at 1 sy-vline.
  write at 2 '일'.
  write at 4 sy-vline.
  write at 5 '이'.
  write at 7  sy-vline.
  write at 8 '삼'.
  write at 10 sy-vline.
endform.                                                  "solution1

*&---------------------------------------------------------------------*
*&      Form  solution2
*&---------------------------------------------------------------------*
form solution2.
  data: text(10) type c.
  text = 'hello'.
  write :/ text right-justified.
  text = '일이삼'.
  write :/ text right-justified.
endform.                                                  "solution2

*&---------------------------------------------------------------------*
*&      Form  solution3
*&---------------------------------------------------------------------*
form solution3.
  data: len      type i,
        text(10) type c.
```

```
  text = 'OTTO'.

  len = cl_abap_list_utilities=>dynamic_output_length( text ).
  write at /(len) text.
  text = '가갸거겨고교구규그기'.
  len = cl_abap_list_utilities=>dynamic_output_length( text ).
  write at /(len) text.
endform.                                                    "solution3

*&---------------------------------------------------------------------*
*&      Form  solution4
*&---------------------------------------------------------------------*
form solution4.
  data: wa         type teched02_numbers,
        line(100)  type c,
        offset_tab type abap_offset_tab,
        pos        type i.

* solution a)
  write :/ 'Solution a)'.
  select * from teched02_numbers into wa order by num lang.
    write:/ wa-lang  no-gap.
    write   sy-vline no-gap.
    write   wa-name  no-gap.
    write   sy-vline no-gap.
    write   (3) wa-num right-justified.
  endselect.

* solution b)
  skip.
  write :/ 'Solution b)'.
  append 3 to offset_tab.
  append 8 to offset_tab.
  select * from teched02_numbers into wa order by num lang.
    write wa-lang  to line(2).
    write sy-vline to line+2(1).
    write wa-name  to line+3(5).
    write sy-vline to line+8(1).
    write wa-num   to line+9(3) right-justified.
    call method cl_abap_list_utilities=>memory_to_display
      exporting
        memory_data    = line
        offset_tab     = offset_tab
      importing
        display_data   = line
        truncation_pos = pos.
    if pos <> -1.
      write :/ 'Truncation is done: '.                      "#EC *
    endif.
    write :/ line.
  endselect.
endform.                                                    "solution5

*&---------------------------------------------------------------------
*&      Form  head
*&---------------------------------------------------------------------
form head  using text type clike.
  skip.
  write :/ sy-uline.
  write :/ text.
  write :/ sy-uline.
  skip.
endform.                    " head
```

Solution a) - Exercise 9

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_9a
*&---------------------------------------------------------------------*
report teched_unicode_solution_9a.

  selection-screen comment 1(80) head1.
  selection-screen skip.
  selection-screen comment 1(80) head2.
  selection-screen skip.
  selection-screen skip.
  selection-screen comment 1(80) lineal1.
  selection-screen skip.
  selection-screen comment 1(80) lineal2.
  selection-screen skip.
  selection-screen comment 1(80) text1.
  selection-screen skip.
  selection-screen comment 1(80) text2.
  selection-screen skip.
  selection-screen comment 1(80) text3.
  selection-screen skip.
  selection-screen comment 1(80) text4.
  selection-screen skip.
  selection-screen begin of line.
  selection-screen comment 1(17) sctext.
  selection-screen position 19.
  parameters: scrolcol type i default 15.
  selection-screen end of line.

  types t_line(100) type c.
  data: line type t_line,
        tab  type table of t_line.

  head1 = 'Please use a Korean Font to look at the list!'.
  head2 = 'Solution a): SCROLL LIST TO COLUMN'.
  lineal1 = '....+....1....+....2....+....3'.
  lineal2 = '12345678901234567890123456789O'.
  text1 = '이름1: xxxxxx 이름2: yyyyyy'.
  text2 = '이름1: 남명희 이름2: 조홍우'.
  text3 = 'Name1: Schell Name2: Fuchs '.  "#EC NOTEXT
  text4 = 'Name1: Mayer  Name2: Novak '.  "#EC NOTEXT
  sctext = 'Scroll to column:'.           "#EC NOTEXT
  append: lineal1 to tab,
          lineal2 to tab,
          text1   to tab,
          text2   to tab,
          text3   to tab,
          text4   to tab.


start-of-selection.

  loop at tab into line.
    write:/ line.
  endloop.
  scroll list to column scrolcol.
```

Solution b) - Exercise 9

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_9b
*&---------------------------------------------------------------------*
report teched_unicode_solution_9b.

  selection-screen comment 1(80) head1.
  selection-screen skip.
  selection-screen comment 1(80) head2.
  selection-screen skip.
  selection-screen skip.
  selection-screen comment 1(80) lineal1.
  selection-screen skip.
  selection-screen comment 1(80) lineal2.
  selection-screen skip.
  selection-screen comment 1(80) text1.
  selection-screen skip.
  selection-screen comment 1(80) text2.
  selection-screen skip.
  selection-screen comment 1(80) text3.
  selection-screen skip.
  selection-screen comment 1(80) text4.
  selection-screen skip.
  selection-screen begin of line.
  selection-screen comment 1(17) sctext.
  selection-screen position 19.
  parameters: scrolcol type i default 15.
  selection-screen end of line.

  types t_line(100) type c.
  data: line    type t_line,
        tab     type table of t_line,
        dis_off type i,
        mem_off type i.

  head1 = 'Please use a Korean Font to look at the list!'.  "#EC NOTEXT
  head2 = 'Solution b): Calculate memory lengths'.          "#EC NOTEXT
  lineal1 = '....+....1....+....2....+....3'.
  lineal2 = '12345678901234567890123456789O'.
  text1 =   '이름1: xxxxxx 이름2: yyyyyy'.
  text2 =   '이름1: 남명희 이름2: 조홍우'.
  text3 = 'Name1: Schell Name2: Fuchs '.                    "#EC NOTEXT
  text4 = 'Name1: Mayer  Name2: Novak '.                    "#EC NOTEXT
  sctext = 'Scroll to column:'.                             "#EC NOTEXT
  append: lineal1 to tab,
          lineal2 to tab,
          text1   to tab,
          text2   to tab,
          text3   to tab,
          text4   to tab.


start-of-selection.

  dis_off = scrolcol - 1.
  loop at tab into line.
    call method cl_abap_list_utilities=>memory_offset
      exporting
        field          = line
```

```
        display_offset = dis_off
      importing
        memory_offset  = mem_off.
    write:/ line+mem_off.
  endloop.
```

Solution Exercise 10

```
*&---------------------------------------------------------------------*
*& Report  TECHED_UNICODE_SOLUTION_10
*&---------------------------------------------------------------------*
report teched_unicode_solution_10.

initialization.
  data: text10_1 type string.
  import text to text10_1 from database teched02_unicode(ar) id 'text10_1'.

at line-selection.

  data: dis_off  type i,
        mem_off  type i,
        dis_col  type i,
        mem_col  type i,
        f        type string. "#EC NEEDED

  get cursor field f memory  offset mem_off.
  get cursor field f display offset dis_off.
  mem_col = mem_off + 1.
  dis_col = dis_off + 1.

  sy-lisel+mem_off(1) = 'x'.
  modify current line.

  read line 10 of current page.
  write dis_col to sy-lisel+16(10).
  modify line 10 of current page.

  read line 11 of current page.
  write mem_col to sy-lisel+16(10).
  modify line 11 of current page.

start-of-selection.

  data: text(100) type c.

  write :/ 'Please use a Korean Font to look at the list!'. "#EC NOTEXT
  skip.

  write :/ 'Please replace all o with x by double clicking on it:'.
                                                          "#EC NOTEXT
  skip.
  text = 'aa->o<-bbcc->o<-ddeeff->o<-gghhiijj'.              "#EC NOTEXT
  write /(50) text.
  text = text10_1.  "'가->o<-갸거->o<-겨고교->o<-구규그기'.
  write /(50) text.
  skip.
  write / 'Display Column:'.                  "#EC NOTEXT
  write / 'Memory  Column:'.                  "#EC NOTEXT
```