SAP ABAP Programozás Alapjai

Első gyakorlat feladatainak egy lehetséges megoldása

A kiadott jegyzetben a feladatok egy lehetséges megoldását mutatom be. A feladat megoldásai során a gyakorlaton használt elnevezési koncepciótól eltértem, de az könnyen leképezhető a hallgató által használtra.

A megoldások ismertetése során a forráskódot angol változónevekkel és angol nyelvű kommentekkel láttam el. A gyakorlat felépítéséhez hasonlóan a feladatok egymásre épülése miatt a magyarázó kommenteket is egymásra épülve ismertetem, azaz egy már ismert magyarázatot nem feltétlenül ismertetek még egyszer egy nagyobb számú feladatnál.

Minden megoldás során az osztály forráskódját és a konzolra írt kimenetet ismertetem!

Tartalomjegyzék

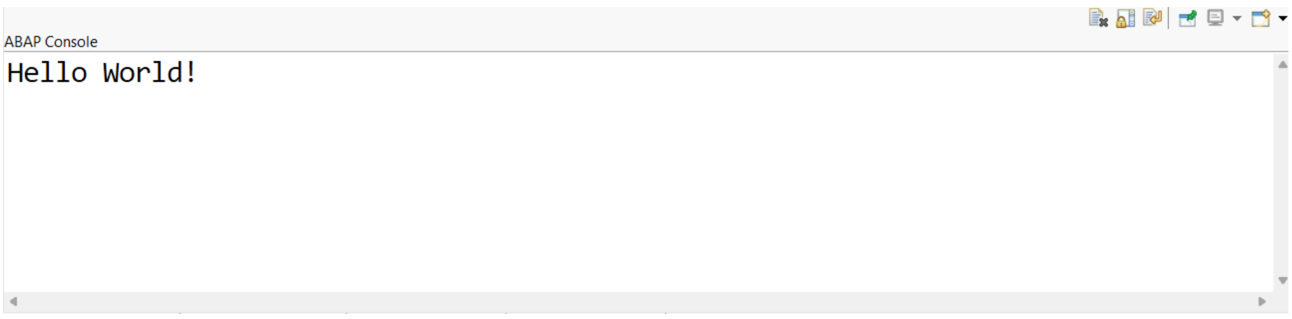
0. Feladat – Hello World!

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f0 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    " Required interface for ABAP Development Tools (ADT)
    " To declare ABAP class as a console application
    " and provide the out parameter
    INTERFACES if_oo_adt_classrun .

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f0 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.
    " out is a parameter and the write method is used to
    " provide data for the console
    " the WRITE ABAP statement is not usable in ABAP on Cloud version
    out->write( 'Hello World!' ).

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

Hello World!



```
Hello World!
```

1. Feladat – Beépített típusok, egyszerű matematikai műveletek

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f1 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f1 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    " Define local variables

    " First define 3 integers without the : , short format.
    " Two integers has defined initial value.
    DATA integer_1 TYPE i VALUE 3.
    DATA integer_2 TYPE i VALUE 5.
    DATA integer_3 TYPE i.

    " Define two packed number with the : , short format
    DATA: packed_number_1 TYPE p LENGTH 8 DECIMALS 3,
          packed_number_2 TYPE p LENGTH 8 DECIMALS 3 VALUE '11.12',
          packed_number_3 TYPE p LENGTH 8 DECIMALS 3.

    " Define character string with initial value
    " Take care that character literal is defined with ' '
    DATA fix_sized_text_1 TYPE c LENGTH 20 VALUE 'fixed sized text'.

    " Define string with initial value
    " Take care that string literal is defined with ` `
    DATA string_1 TYPE string VALUE `Rock 'n roll`.

    " Define datum and time without initial value
    DATA: datum TYPE d,
          tim   TYPE t.

    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    " Working with integer values
    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    integer_3 = 5.

    out->write( 'Basic operations with integers:' ).
    out->write( | { integer_1 } + { integer_2 } = { integer_1 +
integer_2 } | ).
    out->write( | { integer_1 } * { integer_2 } = { integer_1 *
integer_2 } | ).
    out->write( | { integer_1 } / { integer_2 } = { integer_1 /
integer_2 } (!!!)| ).
    out->write( | { integer_2 } / { integer_1 } = { integer_2 /
integer_1 } (!!!)| ).

    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    " Working with packed numbers
    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    packed_number_1 = '5.1'.

    out->write( |\n| ).
    out->write( 'Basic operations with packed numbers:' ).
    out->write( | { packed_number_1 } + { packed_number_2 } =
{ packed_number_1 + packed_number_2 } | ).

    packed_number_3 = packed_number_1 * packed_number_2.
    out->write( | { packed_number_1 } * { packed_number_2 } =
{ packed_number_3 } (!!! Result calculated to a dedicated variable)| ).
    out->write( | { packed_number_1 } * { packed_number_2 } =
{ packed_number_1 * packed_number_2 } (!!! Result calculated as string
template and local expression)| ).

    packed_number_3 = packed_number_1 / packed_number_2.
    out->write( | { packed_number_1 } / { packed_number_2 } =
{ packed_number_3 } (!!! Result calculated to a dedicated variable)| ).
    out->write( | { packed_number_1 } / { packed_number_2 } =
{ packed_number_1 / packed_number_2 } (!!! Result calculated as string
template and local expression)| ).

    packed_number_3 = packed_number_2 / packed_number_1.
    out->write( | { packed_number_2 } / { packed_number_1 } =
{ packed_number_3 } (!!! Result calculated to a dedicated variable)| ).
    out->write( | { packed_number_2 } / { packed_number_1 } =
{ packed_number_2 / packed_number_1 } (!!! Result calculated as string
template and local expression)| ).

    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    " Working with texts
    """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    out->write( |\n| ).
    out->write( | { fix_sized_text_1 } | ).
    out->write( | { string_1 } | ).

    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    " Working with date and time
    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    out->write( |\n| ).
    out->write( | { datum } | ).
    out->write( | { tim } | ).

    " Get actual data and time in ABAP on cloud
    " Universal ABAP used sy-datum and sy-timlo for the same
    datum = cl_abap_context_info=>get_system_date( ).
    tim = cl_abap_context_info=>get_system_time( ).

    " Write date and time in internal format
    out->write( |\n| ).
    out->write( | { datum } | ).
    out->write( | { tim } | ).

    " Write date and time in ISO format
    out->write( |\n| ).
    out->write( | { datum DATE = ISO } | ).
    out->write( | { tim TIME = ISO } | ).

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

```
Basic operations with integers:
 3 + 5 = 8
 3 * 5 = 15
 3 / 5 = 1 (!!!)
 5 / 3 = 2 (!!!)


Basic operations with packed numbers:
 5.100 + 11.120 = 16.220
 5.100 * 11.120 = 56.712 (!!! Result calculated to a dedicated variable)
 5.100 * 11.120 = 56.71200 (!!! Result calculated as string template and local expression)
 5.100 / 11.120 = 0.459 (!!! Result calculated to a dedicated variable)
 5.100 / 11.120 = 0.45863309352518 (!!! Result calculated as string template and local expression)
```

11.120 / 5.100 = 2.180 (!!! Result calculated to a dedicated variable)

11.120 / 5.100 = 2.18039215686275 (!!! Result calculated as string template and local expression)

fixed sized text

Rock 'n roll

00000000

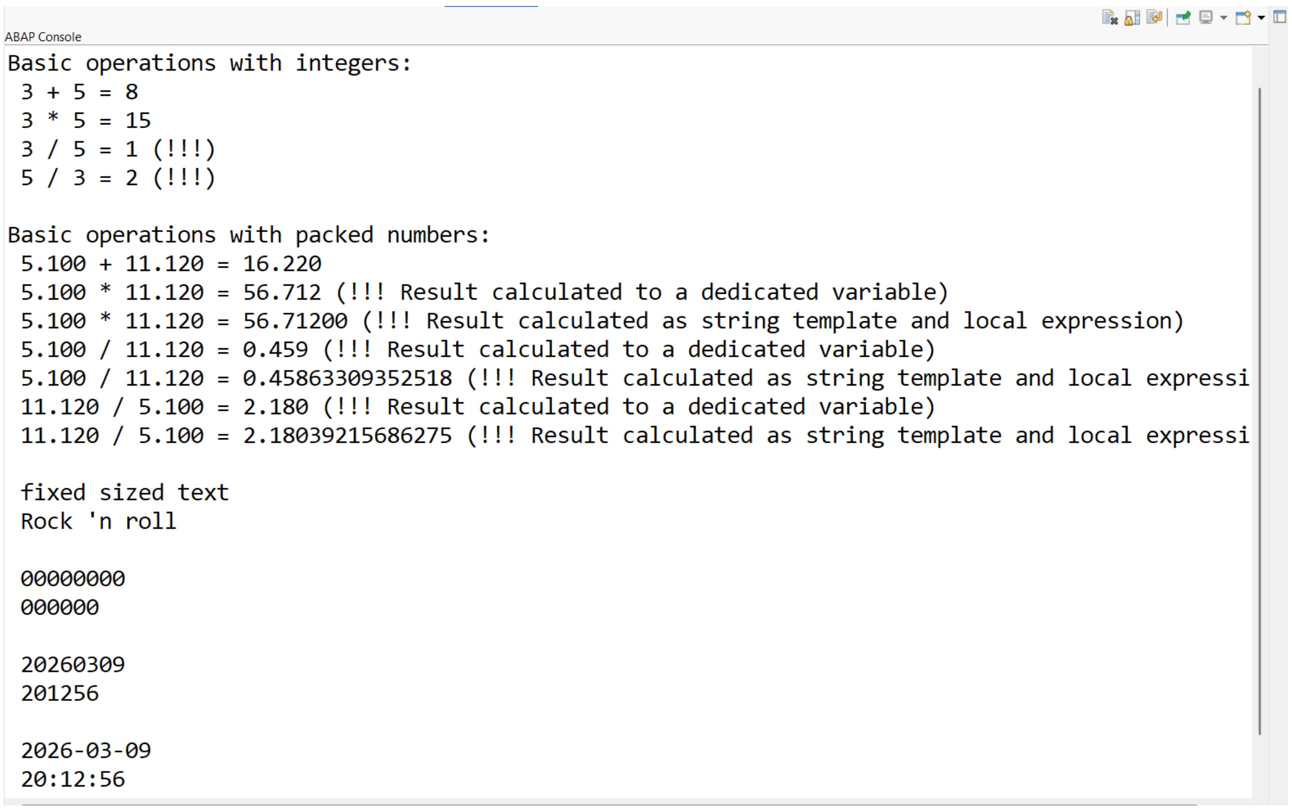
000000

20260309

201256

2026-03-09

20:12:56



```
Basic operations with integers:
 3 + 5 = 8
 3 * 5 = 15
 3 / 5 = 1 (!!!)
 5 / 3 = 2 (!!!)

Basic operations with packed numbers:
 5.100 + 11.120 = 16.220
 5.100 * 11.120 = 56.712 (!!! Result calculated to a dedicated variable)
 5.100 * 11.120 = 56.71200 (!!! Result calculated as string template and local expression)
 5.100 / 11.120 = 0.459 (!!! Result calculated to a dedicated variable)
 5.100 / 11.120 = 0.45863309352518 (!!! Result calculated as string template and local expressi
 11.120 / 5.100 = 2.180 (!!! Result calculated to a dedicated variable)
 11.120 / 5.100 = 2.18039215686275 (!!! Result calculated as string template and local expressi

 fixed sized text
 Rock 'n roll

 00000000
 000000

 20260309
 201256

 2026-03-09
 20:12:56
```

2. Feladat – Egészekből álló belső táblázat és egyszerű műveletvégzések

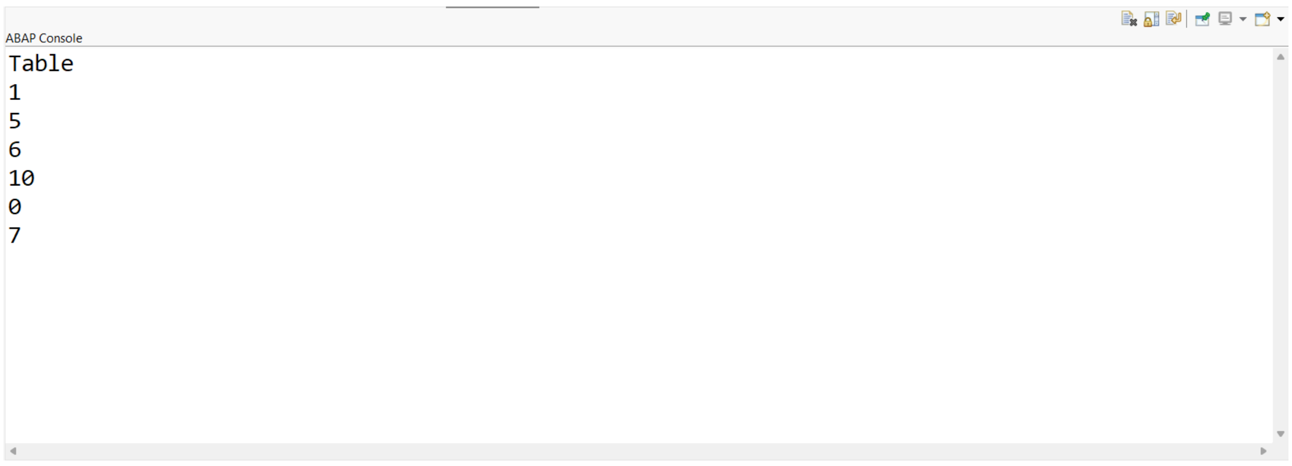
Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f2 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f2 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    " Local definition of an internal table
    " Row type is built-in integer type
    DATA numbers TYPE TABLE OF i.

    " Appending one value to the table
    APPEND  1 TO numbers.

    " Appending more values with : , option
    " This is equivalent with 3 different APPEND statements
    APPEND: 5  TO numbers,
            6  TO numbers,
            10 TO numbers.

    " Appending an initial line to the table
    " As row type is i, the initial value is 0
    APPEND INITIAL LINE TO numbers.

    " Inserting a record to the table
    " As table is a STANDARD table, behaviour of INSERT
    " is equivalent with APPEND statement
    INSERT 7 INTO TABLE numbers.

    " Writing table content to the console
    out->write( numbers ).

  ENDMETHOD.
```

```
ENDCLASS.
```

A konzolra írt kimenet

```
Table
1
5
6
10
0
7
```



```
Table
1
5
6
10
0
7
```

3. Feladat – Stringekből álló belső táblázat és a VALUE constructor

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f3 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f3 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    DATA texts TYPE TABLE OF string.

    " Instead of using APPEND / INSERT statements
    " VALUE constructor is used to declare value of the internal table
    " The statement overwrites the existing content - if any
    " Take care that ` ` is used to declare a string literal

    texts = VALUE #(                    " First ( initiates the start of
the internal table
                     ( `Monday` )       " The first record ( )
                     ( `Tuesday` )      " The second
                     ( `Wednesday` )    " ..
                     ( `Thursday` )
                     ( `Friday` )
                   ).                   " The last ) closes the internal
table

    out->write( texts ).

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

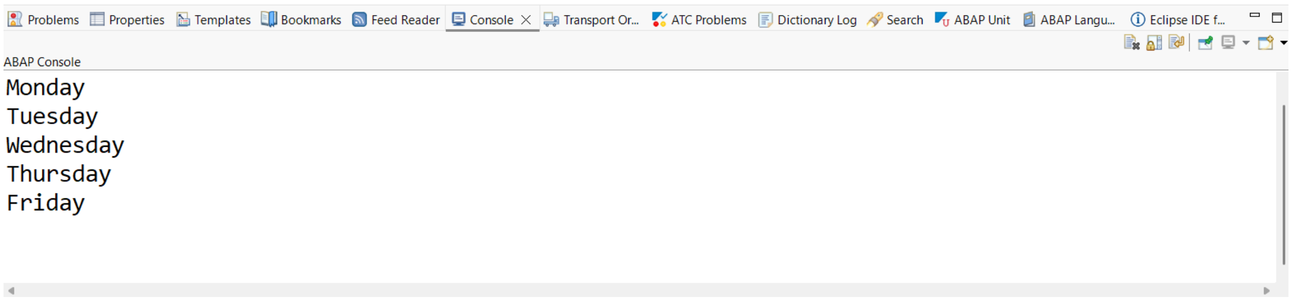
Monday

Tuesday

Wednesday

Thursday

Friday



```
Monday
Tuesday
Wednesday
Thursday
Friday
```

4. Feladat – DO and WHILE iteration

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f4 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f4 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    DATA numbers TYPE TABLE OF i.

    " Do a loop with 5 hard-coded iteration
    DO 5 TIMES.
      " Use system variable sy-index
      " Take care that sy-index is counting from 1
      APPEND sy-index * 2 TO numbers.
    ENDDO.

    " Check content of the internal table
    out->write( numbers ).

    out->write( | \n Read predefined indexes from the table | ).

    " This iteration is an "artificial" iteration
    " just to present infinite loop and
    " accessing content of an internal table with index
    " if index based access is supported

    WHILE 1 = 1.    " This is an infinite loop as this expression is
always true

      " Hard-coded exit from the loop
      " lines( ) returns the actual number of records in the internal
table
      IF sy-index > lines( numbers ).
        EXIT.           " EXIT statement is breaking the WHILE loop
      ENDIF.
      " Get predefined indexes from the table
      CASE sy-index.
        WHEN 2 OR 3 OR 5.
          out->write( | { sy-index }. member of the table is
{ numbers[ sy-index ] } | ).
      ENDCASE.
    ENDWHILE.

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

Table

2

4

6

8

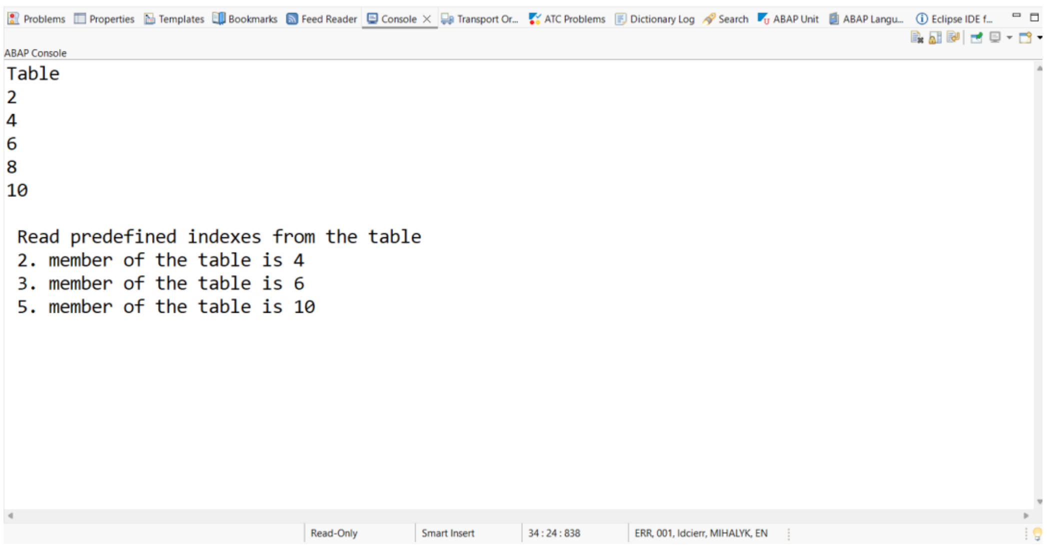
10

Read predefined indexes from the table

2. member of the table is 4

3. member of the table is 6

5. member of the table is 10



```
Table
2
4
6
8
10

 Read predefined indexes from the table
 2. member of the table is 4
 3. member of the table is 6
 5. member of the table is 10
```

5. Feladat – Local structure, sorted table

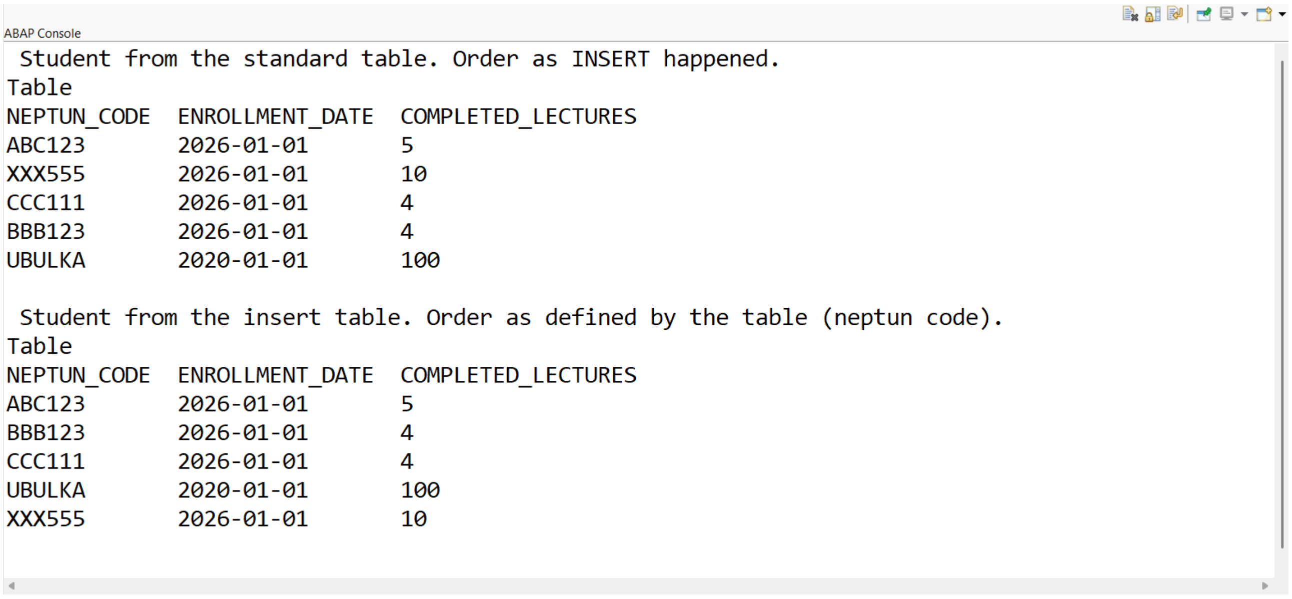
Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f5 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f5 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    TYPES: BEGIN OF student_data,
             neptun_code        TYPE c LENGTH 6,
             enrollment_date    TYPE d,
             completed_lectures TYPE i,
           END OF student_data.

    DATA student_record TYPE student_data.
    DATA student_records_standard TYPE TABLE OF student_data.
    DATA student_records_sorted   TYPE SORTED TABLE OF student_data WITH
UNIQUE KEY neptun_code.

    student_record-neptun_code = 'ABC123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 5.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'XXX555'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 10.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'CCC111'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.   " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'BBB123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.   " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'UBULKA'.
    student_record-enrollment_date = '20200101'.
    student_record-completed_lectures = 100.
    INSERT student_record INTO TABLE student_records_standard.   " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    out->write( | Student from the standard table. Order as INSERT
happened. | ).
    out->write( student_records_standard ).
    out->write( |\n| ).
    out->write( | Student from the insert table. Order as defined by the
table (neptun code). | ).
    out->write( student_records_sorted ).

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

```
Student from the standard table. Order as INSERT happened.

Table

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES

ABC123      2026-01-01      5

XXX555      2026-01-01      10

CCC111      2026-01-01      4

BBB123      2026-01-01      4

UBULKA      2020-01-01      100


 Student from the insert table. Order as defined by the table (neptun code).

Table
```

NEPTUN_CODE	ENROLLMENT_DATE	COMPLETED_LECTURES
ABC123	2026-01-01	5
BBB123	2026-01-01	4
CCC111	2026-01-01	4
UBULKA	2020-01-01	100
XXX555	2026-01-01	10



```
 Student from the standard table. Order as INSERT happened.
Table
NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES
ABC123       2026-01-01       5
XXX555       2026-01-01       10
CCC111       2026-01-01       4
BBB123       2026-01-01       4
UBULKA       2020-01-01       100

 Student from the insert table. Order as defined by the table (neptun code).
Table
NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES
ABC123       2026-01-01       5
BBB123       2026-01-01       4
CCC111       2026-01-01       4
UBULKA       2020-01-01       100
XXX555       2026-01-01       10
```

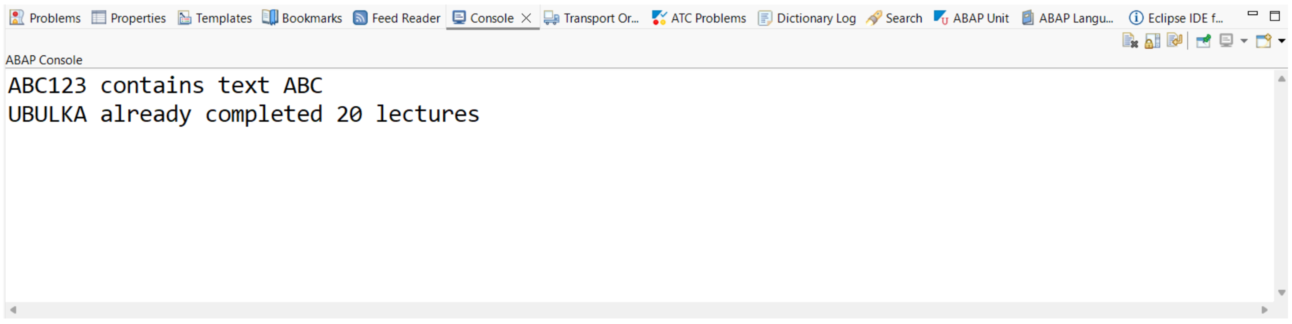
6. Feladat – LOOP over table and check for condition (IF)

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f6 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f6 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    " First part is the same as exercise 5
    TYPES: BEGIN OF student_data,
             neptun_code        TYPE c LENGTH 6,
             enrollment_date    TYPE d,
             completed_lectures TYPE i,
           END OF student_data.

    DATA student_record TYPE student_data.
    DATA student_records_standard TYPE TABLE OF student_data.
    DATA student_records_sorted   TYPE SORTED TABLE OF student_data WITH
UNIQUE KEY neptun_code.

    student_record-neptun_code = 'ABC123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 5.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'XXX555'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 10.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'CCC111'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'BBB123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'UBULKA'.
    student_record-enrollment_date = '20200101'.
    student_record-completed_lectures = 100.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    " Loop over the internal table (yet without WHERE condition)
    " and write out data if a condition is met
    " Take care that both the standard and sorted table can be used
    LOOP AT student_records_standard INTO student_record.

        " If student completed already 20 lectures -> write out
        IF student_record-completed_lectures >= 20.
            out->write( |{ student_record-neptun_code } already
completed 20 lectures| ).
        ENDIF.

        " If student NEPTUN code contains 'ABC' -> write out
        IF student_record-neptun_code CS 'ABC'.     " Alternative CP
'*ABC*'
            out->write( |{ student_record-neptun_code } contains text
ABC| ).
        ENDIF.

    ENDLOOP.

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

```
ABC123 contains text ABC

UBULKA already completed 20 lectures
```



```
ABC123 contains text ABC
UBULKA already completed 20 lectures
```

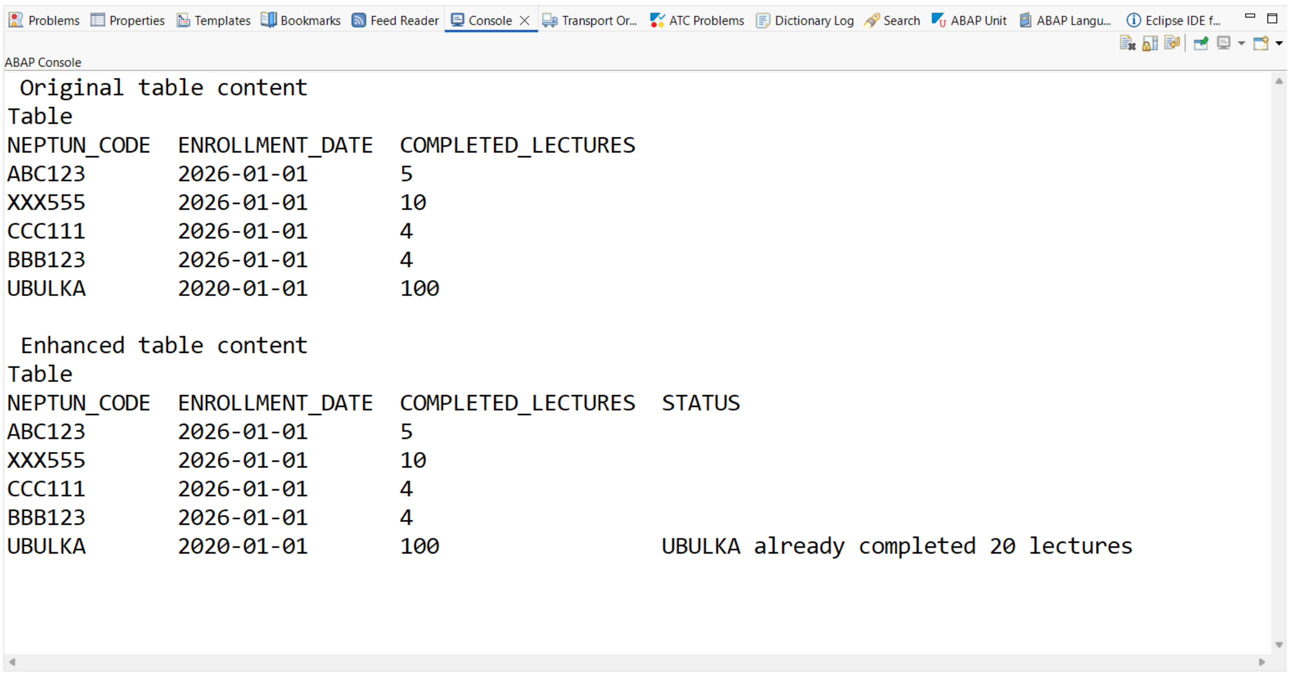
7. Feladat – INCLUDE TYPE and calculation of an additional field

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f7 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_gyak1_f7 IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    " First part is the same as exercise 5
    TYPES: BEGIN OF student_data,
             neptun_code        TYPE c LENGTH 6,
             enrollment_date    TYPE d,
             completed_lectures TYPE i,
           END OF student_data.

    DATA student_record TYPE student_data.
    DATA student_records_standard TYPE TABLE OF student_data.
    DATA student_records_sorted   TYPE SORTED TABLE OF student_data WITH
UNIQUE KEY neptun_code.

    student_record-neptun_code = 'ABC123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 5.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'XXX555'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 10.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'CCC111'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'BBB123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'UBULKA'.
    student_record-enrollment_date = '20200101'.
    student_record-completed_lectures = 100.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    " New part
    " Define local type which includes student data and enhance with a
new field
    " As discussed in the presentation by design flat structure is used
    " Alternative would be a nested structure

    TYPES BEGIN OF student_data_enh.
    INCLUDE TYPE student_data.
    TYPES status TYPE string.
    TYPES END OF student_data_enh.

    DATA: student_record_enh           TYPE student_data_enh,
          student_records_enh_standard TYPE TABLE OF student_data_enh.

    " Calculate enhanced content record-by-record
    " Table data copy or FIELD-SYMBOLS is not yet used

    LOOP AT student_records_standard INTO student_record.

      " Clear enhanced record and move corresponding (!) all data
      CLEAR student_record_enh.
      MOVE-CORRESPONDING student_record TO student_record_enh.

      " Calculate "status" attribute
      IF student_record-completed_lectures >= 20.
        student_record_enh-status = |{ student_record-neptun_code }
already completed 20 lectures|.
      ENDIF.
      " Insert new record to the enhanced table
      INSERT student_record_enh INTO TABLE student_records_enh_standard.
    ENDLOOP.

    " Write out original and enhanced tables
    out->write( | Original table content | ).
    out->write( student_records_standard ).
    out->write( | \n Enhanced table content | ).
    out->write( student_records_enh_standard ).

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

```
Original table content

Table

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES

ABC123       2026-01-01       5

XXX555       2026-01-01       10

CCC111       2026-01-01       4

BBB123       2026-01-01       4

UBULKA       2020-01-01       100


 Enhanced table content

Table

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS

ABC123       2026-01-01       5

XXX555       2026-01-01       10

CCC111       2026-01-01       4

BBB123       2026-01-01       4

UBULKA       2020-01-01       100                 UBULKA already completed 20 lectures
```



```
 Original table content
Table
NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES
ABC123       2026-01-01       5
XXX555       2026-01-01       10
CCC111       2026-01-01       4
BBB123       2026-01-01       4
UBULKA       2020-01-01       100

 Enhanced table content
Table
NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS
ABC123       2026-01-01       5
XXX555       2026-01-01       10
CCC111       2026-01-01       4
BBB123       2026-01-01       4
UBULKA       2020-01-01       100                 UBULKA already completed 20 lectures
```

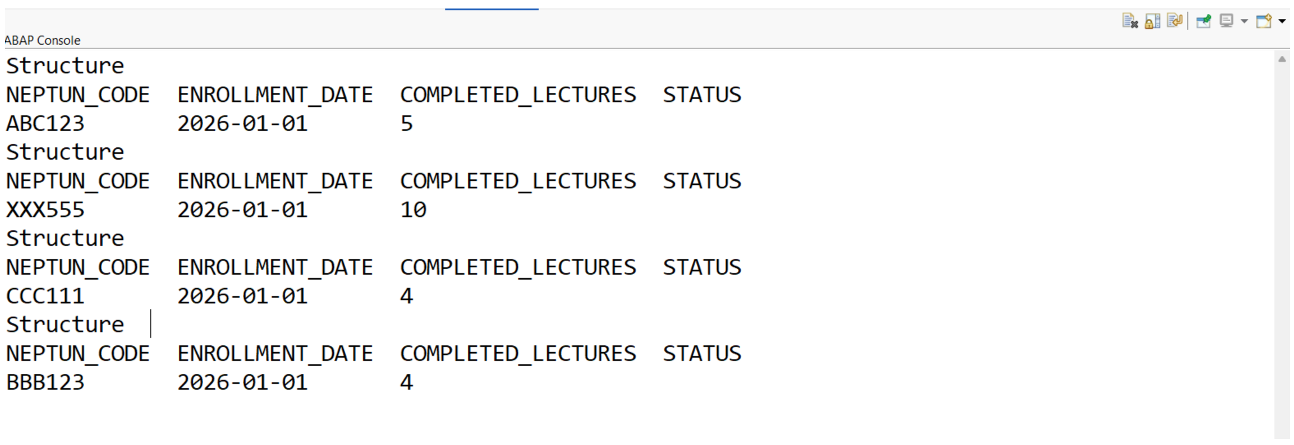
8. Feladat – LOOP ciklus WHERE használatával

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f8 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS zmk_cl_uni_2026_gyak1_f8 IMPLEMENTATION.


  METHOD if_oo_adt_classrun~main.

    " First part is the same as exercise 5
    TYPES: BEGIN OF student_data,
             neptun_code         TYPE c LENGTH 6,
             enrollment_date     TYPE d,
             completed_lectures TYPE i,
           END OF student_data.

    DATA student_record TYPE student_data.
    DATA student_records_standard TYPE TABLE OF student_data.
    DATA student_records_sorted   TYPE SORTED TABLE OF student_data WITH
UNIQUE KEY neptun_code.

    student_record-neptun_code = 'ABC123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 5.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'XXX555'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 10.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.
    student_record-neptun_code = 'CCC111'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'BBB123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'UBULKA'.
    student_record-enrollment_date = '20200101'.
    student_record-completed_lectures = 100.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    " New part
    " Define local type which includes student data and enhance with a
new field
    " As discussed in the presentation by design flat structure is used
    " Alternative would be a nested structure

    TYPES BEGIN OF student_data_enh.
    INCLUDE TYPE student_data.
    TYPES status TYPE string.
    TYPES END OF student_data_enh.

    DATA: student_record_enh           TYPE student_data_enh,
          student_records_enh_standard TYPE TABLE OF student_data_enh.

    LOOP AT student_records_standard INTO student_record.
      CLEAR student_record_enh.
      MOVE-CORRESPONDING student_record TO student_record_enh.

      IF student_record-completed_lectures >= 20.
        student_record_enh-status = |{ student_record-neptun_code }
already completed 20 lectures|.
      ENDIF.

      INSERT student_record_enh INTO TABLE student_records_enh_standard.
    ENDLOOP.
    " New part
    " Write out records where status field is empty (initial)
    LOOP AT student_records_enh_standard INTO student_record_enh
      WHERE status IS INITIAL.
      out->write( student_record_enh ).
    ENDLOOP.

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet

```
Structure

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS

ABC123     2026-01-01     5

Structure

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS

XXX555     2026-01-01     10

Structure

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS

CCC111     2026-01-01     4

Structure

NEPTUN_CODE  ENROLLMENT_DATE  COMPLETED_LECTURES  STATUS

BBB123     2026-01-01     4
```



```
Structure
NEPTUN_CODE   ENROLLMENT_DATE   COMPLETED_LECTURES   STATUS
ABC123        2026-01-01        5
Structure
NEPTUN_CODE   ENROLLMENT_DATE   COMPLETED_LECTURES   STATUS
XXX555        2026-01-01        10
Structure
NEPTUN_CODE   ENROLLMENT_DATE   COMPLETED_LECTURES   STATUS
CCC111        2026-01-01        4
Structure
NEPTUN_CODE   ENROLLMENT_DATE   COMPLETED_LECTURES   STATUS
BBB123        2026-01-01        4
```

9. Feladat – LOOP ciklus WHERE használatával

Megoldás forráskódja

```
CLASS zmk_cl_uni_2026_gyak1_f9 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS zmk_cl_uni_2026_gyak1_f9 IMPLEMENTATION.


  METHOD if_oo_adt_classrun~main.
    " First part is the same as exercise 5
    TYPES: BEGIN OF student_data,
             neptun_code        TYPE c LENGTH 6,
             enrollment_date    TYPE d,
             completed_lectures TYPE i,
           END OF student_data.

    DATA student_record TYPE student_data.
    DATA student_records_standard TYPE TABLE OF student_data.
    DATA student_records_sorted   TYPE SORTED TABLE OF student_data WITH
UNIQUE KEY neptun_code.

    student_record-neptun_code = 'ABC123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 5.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'XXX555'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 10.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.
    student_record-neptun_code = 'CCC111'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'BBB123'.
    student_record-enrollment_date = '20260101'.
    student_record-completed_lectures = 4.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    student_record-neptun_code = 'UBULKA'.
    student_record-enrollment_date = '20200101'.
    student_record-completed_lectures = 100.
    INSERT student_record INTO TABLE student_records_standard.  " Same
as APPEND would be !
    INSERT student_record INTO TABLE student_records_sorted.

    " New part
    " Define local type which includes student data and enhance with a
new field
    " As discussed in the presentation by design flat structure is used
    " Alternative would be a nested structure

    TYPES BEGIN OF student_data_enh.
    INCLUDE TYPE student_data.
    TYPES status TYPE string.
    TYPES END OF student_data_enh.

    DATA: student_record_enh           TYPE student_data_enh,
          student_records_enh_standard TYPE TABLE OF student_data_enh.

    LOOP AT student_records_standard INTO student_record.
      CLEAR student_record_enh.
      MOVE-CORRESPONDING student_record TO student_record_enh.

      IF student_record-completed_lectures >= 20.
        student_record_enh-status = |{ student_record-neptun_code }
already completed 20 lectures|.
      ENDIF.

      INSERT student_record_enh INTO TABLE student_records_enh_standard.
    ENDLOOP.

    " New part
    " Write out records between FROM and TO index
    DATA: lv_index_from TYPE i VALUE 2,
          lv_index_to   TYPE i VALUE 5.

    LOOP AT student_records_enh_standard INTO student_record_enh
      FROM lv_index_from
        TO lv_index_to.
      out->write( | In the enhanced table { sy-tabix }. student neptun
code is { student_record_enh-neptun_code } | ).
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

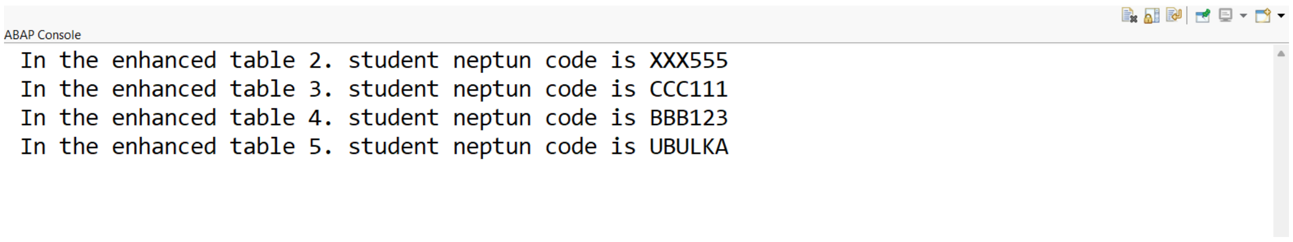
A konzolra írt kimenet

In the enhanced table 2. student neptun code is XXX555

In the enhanced table 3. student neptun code is CCC111

In the enhanced table 4. student neptun code is BBB123

In the enhanced table 5. student neptun code is UBULKA



```
 In the enhanced table 2. student neptun code is XXX555
 In the enhanced table 3. student neptun code is CCC111
 In the enhanced table 4. student neptun code is BBB123
 In the enhanced table 5. student neptun code is UBULKA
```