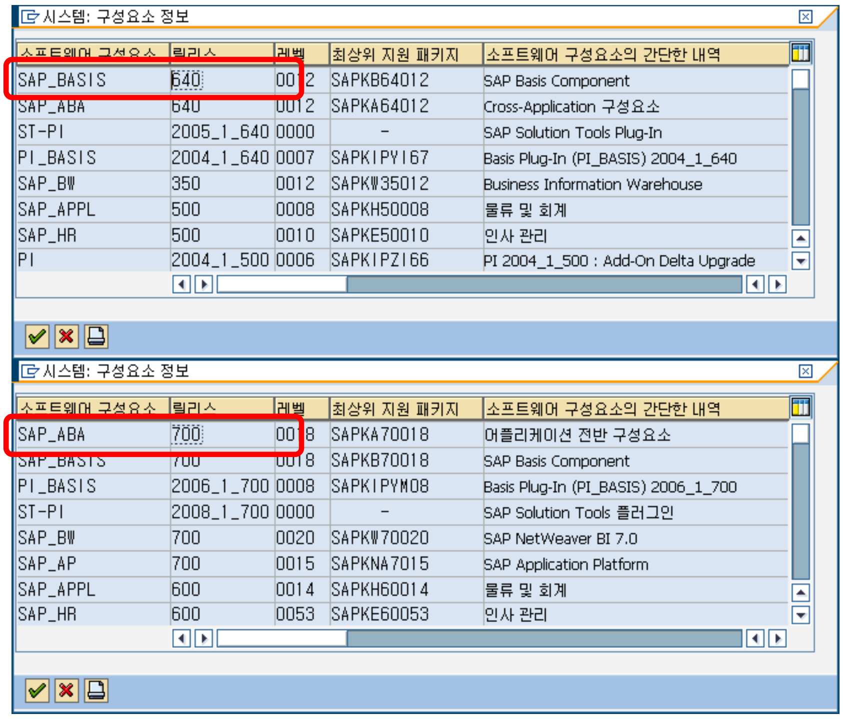
Dynamic Internal table

1. 동적으로 변수 생성하기

SAP Release 640 버전이후부터는 동적으로 구조체 및 인터널 테이블을 생성할 수 있는 RTTS 가 제공된다. 메뉴 : *시스템 -> 상태 -> 구성요소 버전 ->* 을 선택하면 시스템 정보를 확인할 수 있다. [그림 1]과 같이 SAP_BASIS 영역이 640 이상이면 RTTS 를 사용할 수 있다.


시스템: 구성요소 정보

소프트웨어 구성요소	릴리스	레벨	최상위 지원 패키지	소프트웨어 구성요소의 간단한 내역
SAP_BASIS	640	0012	SAPKB64012	SAP Basis Component
SAP_ABA	640	0012	SAPKA64012	Cross-Application 구성요소
ST-PI	2005_1_640	0000	-	SAP Solution Tools Plug-In
PI_BASIS	2004_1_640	0007	SAPKIPYI67	Basis Plug-In (PI_BASIS) 2004_1_640
SAP_BW	350	0012	SAPKW35012	Business Information Warehouse
SAP_APPL	500	0008	SAPKH50008	물류 및 회계
SAP_HR	500	0010	SAPKE50010	인사 관리
PI	2004_1_500	0006	SAPKIPZI66	PI 2004_1_500 : Add-On Delta Upgrade

시스템: 구성요소 정보

소프트웨어 구성요소	릴리스	레벨	최상위 지원 패키지	소프트웨어 구성요소의 간단한 내역
SAP_ABA	700	0018	SAPKA70018	어플리케이션 전반 구성요소
SAP_BASIS	700	0018	SAPKB70018	SAP Basis Component
PI_BASIS	2006_1_700	0008	SAPKIPYM08	Basis Plug-In (PI_BASIS) 2006_1_700
ST-PI	2008_1_700	0000	-	SAP Solution Tools 플러그인
SAP_BW	700	0020	SAPKW70020	SAP NetWeaver BI 7.0
SAP_AP	700	0015	SAPKNA7015	SAP Application Platform
SAP_APPL	600	0014	SAPKH60014	물류 및 회계
SAP_HR	600	0053	SAPKE60053	인사 관리

[그림 1 시스템 정보 확인]

RTTS 를 사용하여 동적인 인터널 테이블을 생성하기 위해서는 필드심볼과 데이터 참조에 대한 사전 지식이 필요하다. 개념에 대한 설명은 생략하고 이 두 가지를 이용해 동적으로 변수를 생성하는 예제를 하나 만들어보자.

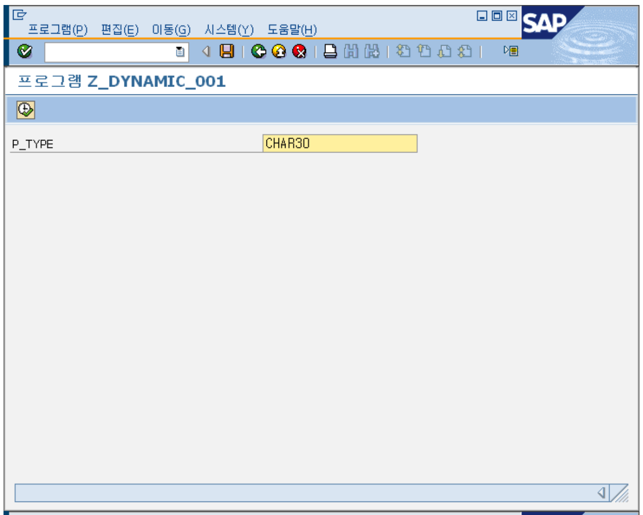
[예제 1]

```
REPORT z_dynamic_01.

PARAMETER p_type(20)  TYPE c.

DATA: dref1 TYPE REF TO data.

FIELD-SYMBOLS: <fs1> TYPE ANY.

CREATE DATA dref1 TYPE (p_type).

ASSIGN dref1->* TO <fs1>.
```

```
<fs1> = 'Dynamic Test'.

WRITE / <fs1>.
```

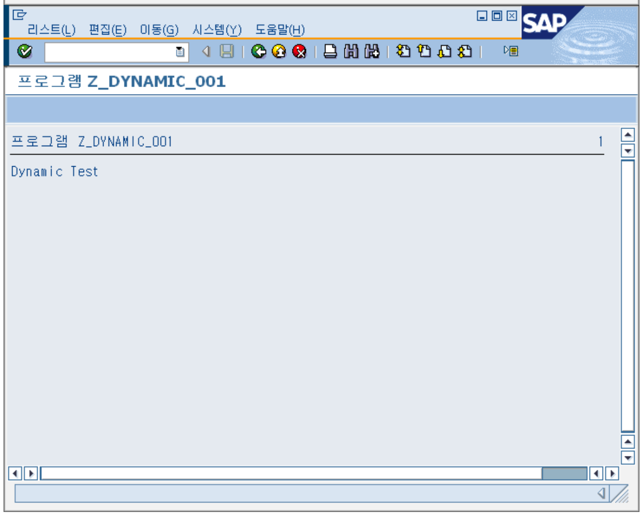
[예제 1] 프로그램을 실행하면 사용자에게 타입을 입력 받아서 동적으로 변수를 생성하게 된다. 즉 [결과 1]에서 알 수 있듯이 CHAR30 자리 수를 입력받아 문자 'Dynamic Test'를 출력하게 된다.

[결과 1]



프로그램 Z_DYNAMIC_001

P_TYPE CHAR30



프로그램 Z_DYNAMIC_001

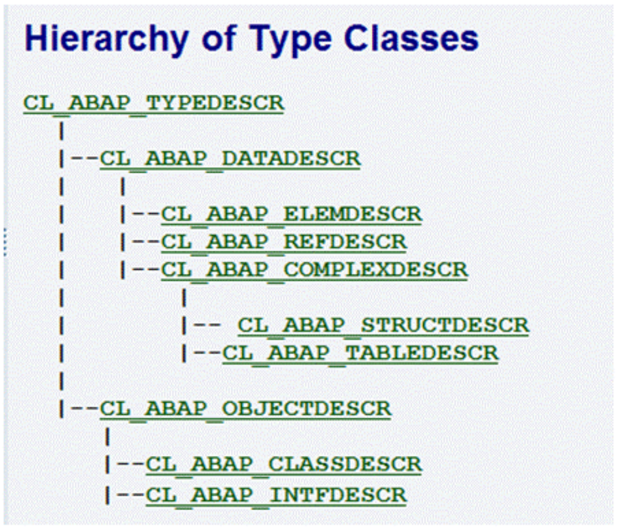
```
프로그램 Z_DYNAMIC_001                                    1

Dynamic Test
```

데이터 참조를 사용하면, CREATE DATA dref1TYPE (p_type) 구문과 같이 데이터 오브젝트 자체를 프로그램 실행 도중에 동적으로 생성하게 된다. 역참조(Dereference)는 프로그래밍 언어에서 프로그램 포인터가 가리키는 번지에 저장된 데이터에 접근하는 것으로 정의된다. 즉, 데이터 참조가 가리키고 있는 데이터 오브젝트의 변수 값에 접근하기 위해서는 Dereference 과정을 거쳐야 한다. 즉, ASSIGN dref1->* TO <fs1>. 구문에서 필드 심볼에 Assign 하는 것을 의미한다.

Dereference(=Assign)가 성공하게 되면 SY-SUBRC 변수는 0 을, 성공하면 4 를 반환한다. '->*' 기호는 Dereferencing Operator 라고 한다. 자세한 사항은 easy abap 서적의 13 장 Field Symbol & Data Reference 를 참고하도록 하자.

2. 동적으로 구조체 생성하기

이번에는 RTTS 를 이용하여 동적으로 구조체를 생성해 보자. 동적인 구조체는 CL_ABAP_STRUCTDESCR 라는 클래스의 CREATE 메소드를 호출하여 구성하게 된다. 이 클래스는 [그림 2]에서 설명하고 있는 TYPE 클래스의 계층구조에 소속된다.

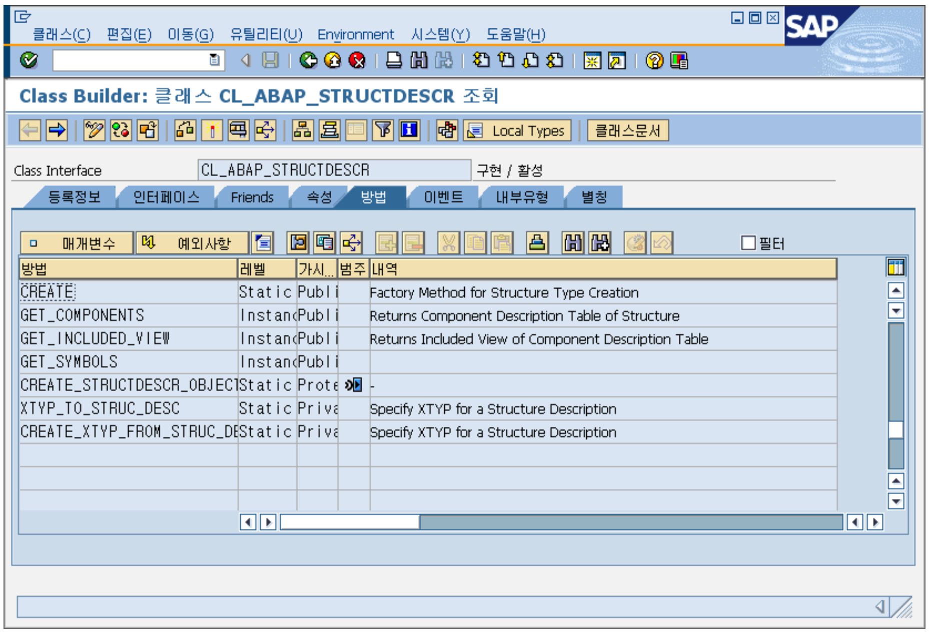


Hierarchy of Type Classes

```
CL_ABAP_TYPEDESCR
  |
  |--CL_ABAP_DATADESCR
  |     |
  |     |--CL_ABAP_ELEMDESCR
  |     |--CL_ABAP_REFDESCR
  |     |--CL_ABAP_COMPLEXDESCR
  |           |
  |           |-- CL_ABAP_STRUCTDESCR
  |           |--CL_ABAP_TABLEDESCR
  |
  |--CL_ABAP_OBJECTDESCR
       |
       |--CL_ABAP_CLASSDESCR
       |--CL_ABAP_INTFDESCR
```

[그림 2 TYPE 클래스 계층 구조]

T-CODE:SE24(Class Builder)에서 CL_ABAP_STRUCTDESCR 클래스를 조회한 후 메소드(방법) 탭으로 이동하자. 그리고 CREATE 메소드를 검색한다.



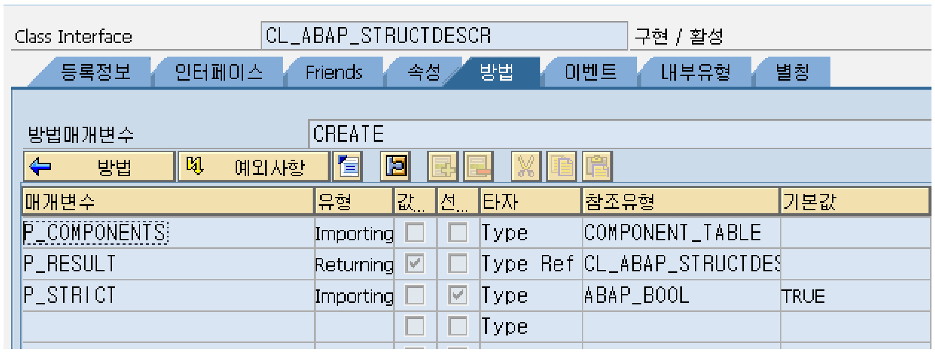
Class Builder: 클래스 CL_ABAP_STRUCTDESCR 조회

Class Interface CL_ABAP_STRUCTDESCR 구현 / 활성

방법	레벨	가시...	범주	내역
CREATE	Static	Publi		Factory Method for Structure Type Creation
GET_COMPONENTS	Instan	Publi		Returns Component Description Table of Structure
GET_INCLUDED_VIEW	Instan	Publi		Returns Included View of Component Description Table
GET_SYMBOLS	Instan	Publi		
CREATE_STRUCTDESCR_OBJEC	Static	Prote		-
XTYP_TO_STRUC_DESC	Static	Priva		Specify XTYP for a Structure Description
CREATE_XTYP_FROM_STRUC_DE	Static	Priva		Specify XTYP for a Structure Description

[그림 3 클래스 빌더에서 CREATE 메소드 조회하기]

상단의 ▫ 매개변수 버튼을 선택하면, 3 개의 파라미터가 존재하는 것을 확인할 수 있다. 첫번째 파라미터 P_COMPONENTS 에 동적으로 생성할 구조체의 필드이름과 데이터 타입을 저장하게 된다.


Class Interface CL_ABAP_STRUCTDESCR 구현 / 활성

방법매개변수 CREATE

매개변수	유형	값..	선..	타자	참조유형	기본값
P_COMPONENTS	Importing	☐	☐	Type	COMPONENT_TABLE	
P_RESULT	Returning	✓	☐	Type Ref	CL_ABAP_STRUCTDES	
P_STRICT	Importing	☐	✓	Type	ABAP_BOOL	TRUE
		☐	☐	Type		

[그림 4 CREATE 메소드 파라미터]

다음 구문과 같이 ABAP 프로그램내에 CREATE 메소드를 호출할 때 파라미터 정보를 전달해야 한다.

```
  CALL METHOD cl_abap_structdescr=>create
    EXPORTING
      p_components = ‘필드 정보’
    RECEIVING
      p_result     = ‘생성할 구조체 오브젝트’
```

위 구문의 축약형(동일한 구문)은 다음과 같다.

```
‘생성할 구조체 오브젝트’
  = cl_abap_structdescr=>create( p_components = ‘필드 정보’).
```

파리미터 P_componets 는 다음 4 개의 필드를 가지는 테이블 타입이다.

필드이름	타입	내역
length	I	필드 길이 TYPE C, P, N 인 경우 자릿수
decimals	I	TYPE P 인 경우 소수자릿수
type_kind	ABAP_TYPEKIND (TYPE C LENGTH 1)	필드 타입
name	ABAP_COMPNAME (TYPE C LENGTH 30)	필드 이름

[표 1 CREATE 메소드의 P_componets 파라미터 정보]

[표 1]을 참고하여 10 자리 문자 TYPE C 인 필드를 구조체에 할당하기 위해서는 CL_ABAP_ELEMDESCR 클래스의 GET_C 메소드를 호출하여 다음과 같이 기술한다. 메소드 이름에서 유추할 수 있듯이 구조체에 필드를 추가하기 위한 필드의 타입은 ‘GET_’ + ABAP 타입으로 정의된 메소드를 호출하면 된다. 즉, TYPE I 는 GET_I, TYPE P 는 GET_P 와 같은 형태이다.

```
CALL METHOD cl_abap_elemdescr=>get_c
  EXPORTING
    p_length = 10
  RECEIVING
    p_result =  ‘리턴’
```

그리고 CL_ABAP_STRUCTDESCR 클래스를 참조하는 데이터 참조 변수를 생성한다. HANDLE 은 RTTS 오브젝트를 참고할 때 추가로 사용하는 구문이다.

```
CREATE DATA lr_wa TYPE HANDLE lr_structdescr.
```

필드 심볼에 ASSIGN 하면 RTTS 를 이용한 구조체가 동적으로 완성된다.

```
ASSIGN lr_wa->* TO <fs_wa>.
```

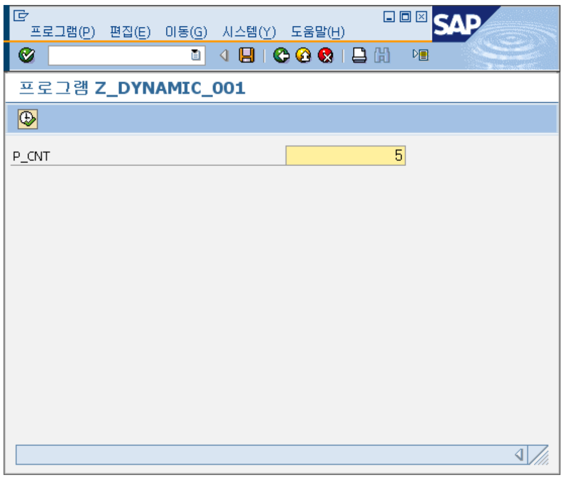
이와 같은 과정으로 구조체를 생성해서 write 구문으로 출력하는 프로그램을 만들어 보자. [예제 2]는 사용자에게 구조체의 필드 개수를 입력 받아서 첫 번째 필드 이름은 column_1 , 두 번째 필드 이름은 column_2 와 같이 정의한 후 숫자값을 할당하여 출력하게 된다.

[예제 2]

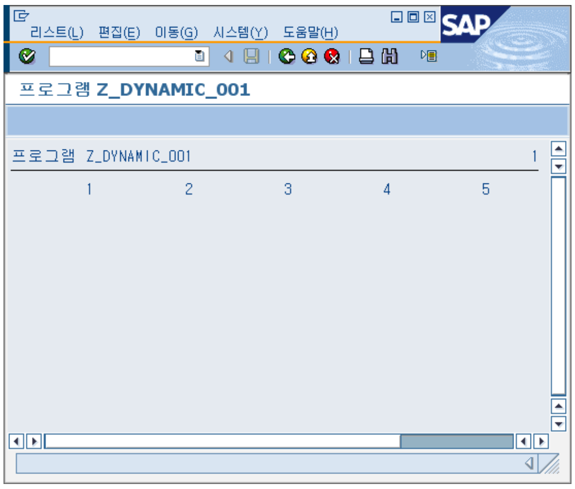
```
REPORT z_dynamic_02.

TYPE-POOLS: abap.

DATA:   lr_structdescr   TYPE REF TO cl_abap_structdescr,
        lr_datadescr     TYPE REF TO cl_abap_datadescr,
        lt_comp          TYPE abap_component_tab,
        ls_comp          TYPE abap_componentdescr,
        lr_wa            TYPE REF TO data.

DATA : lv_idx TYPE n LENGTH 2.

FIELD-SYMBOLS: <fs_field> TYPE ANY.
FIELD-SYMBOLS: <fs_wa>    TYPE ANY.

PARAMETER p_cnt TYPE i.

START-OF-SELECTION.

1 DO p_cnt TIMES.
    lv_idx = lv_idx + 1.
2   CONCATENATE 'column'  lv_idx INTO ls_comp-name
    SEPARATED BY '_'.

3   CALL METHOD cl_abap_elemdescr=>get_i
      RECEIVING
        p_result = ls_comp-type.

4   INSERT ls_comp INTO TABLE lt_comp.
  ENDDO.

5 CALL METHOD cl_abap_structdescr=>create
    EXPORTING
      p_components = lt_comp
    RECEIVING
      p_result     = lr_structdescr.

6 CREATE DATA lr_wa TYPE HANDLE lr_structdescr.
```

```
7  ASSIGN lr_wa->* TO <fs_wa>.

   DO  p_cnt  TIMES.
     DO.
8      ASSIGN COMPONENT  sy-index
              OF STRUCTURE <fs_wa> TO <fs_field>.
       IF sy-subrc NE 0.        EXIT.       ENDIF.
       <fs_field> =  sy-index.
     ENDDO.
   ENDDO.

   DO.
     ASSIGN COMPONENT sy-index
9      OF STRUCTURE <fs_wa> TO <fs_field>.

     IF sy-subrc IS NOT INITIAL. EXIT. ENDIF.
     WRITE  <fs_field>.

   ENDDO.
```

[결과 2]



```
P_CNT    5
```



```
프로그램 Z_DYNAMIC_001                                1

        1         2         3         4         5
```

1. 사용자가 입력한 숫자만큼 필드를 생성하기 위해 DO 구문을 이용해 루프를 수행한다.
2. CONCATENATE 구문을 이용해 구조체 필드의 이름을 COLUMN_1 과 같이 정의한다. sy-index 는 시스템변수로 DO 구문 안에서 수행되는 루프 카운트 정보가 저장되어 있다.
3. 구조체 필드 타입은 GET_I 메소드를 호출하여 정수 I 로 정의한다.
4. 구조체를 생성하는 CREATE 메소드의 파라미터 lt_comp 인터널 테이블에 필드 정보를 추가한다.
5. 구조체 오브젝트를 생성한다.
6. RTTS 오브젝트를 참조하는 lr_wa 데이터 참조 변수를 생성한다.
7. 필드 심볼에 할당한다.
8. ASSIGN COMPONENT 구문을 이용해 구조체 필드 심볼의 개별 필드에 값을 설정한다.
9. 구조체 필드에 저장된 값을 출력하기 위해 필드심볼에 다시 값을 할당한다.

ASSIGN COMPONENT 구문을 이용해 구조체의 개별 필드를 필드 심볼에 ASSIGN 할 수 있다. comp 항목에 올 수 있는 것은 line 의 순번이나 컬럼명이다.

```
ASSIGN COMPONENT comp OF STRUCTURE struc TO <fs>.
```

3. 동적으로 인터널 테이블 생성하기

동적인 인터널 테이블은 구조체 생성과 유사한 방법으로 적용된다. RTTS 를 이용한 인터널테이블은 클래스 **CL_ABAP_TABLEDESCR** 를 이용하여 생성한다. 오브젝트를 생성하는 CREATE 메소드를 호출할 때 [예제 2]에서 생성한 RTTS 구조체 타입을 파라미터로 전달하면 인터털 테이블이 동적으로 구성된다.

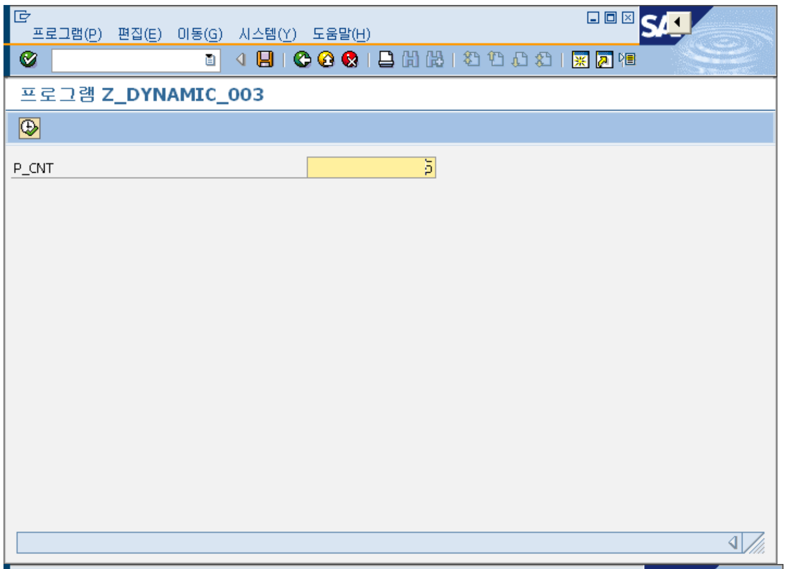
```
lr_datadescr = lr_structdescr.

CALL METHOD cl_abap_tabledescr=>create
      EXPORTING
        p_line_type  =  lr_datadescr
      RECEIVING
        p_result     =  lr_tabledescr .
```

이외 과정은 [예제 2]와 유사하므로, Z_DYNAMIC_02 프로그램을 Z_DYNAMIC_03 으로 복사한 후 BOLD 채로 설정된 라인만 추가해 보자.

[예제 3]

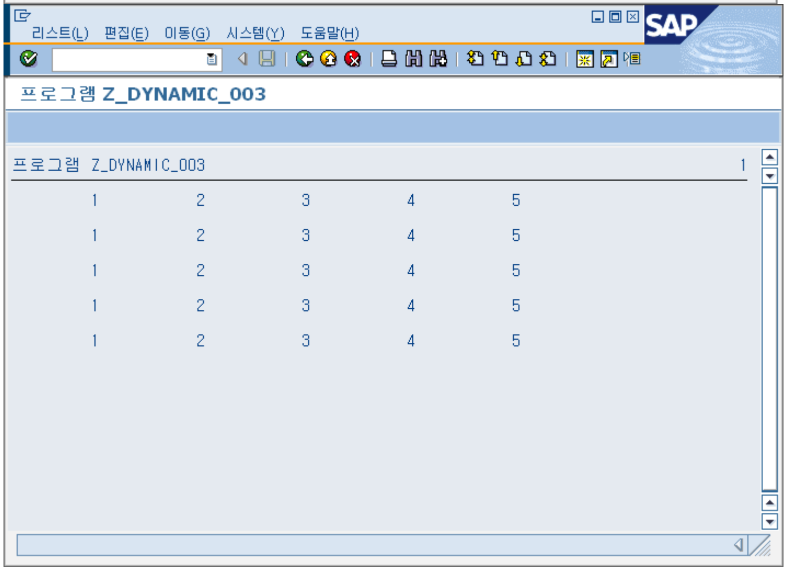
```
REPORT z_dynamic_03.

TYPE-POOLS: abap.

DATA:   lr_structdescr   TYPE REF TO cl_abap_structdescr,
        lr_tabledescr    TYPE REF TO cl_abap_tabledescr,
        lr_datadescr     TYPE REF TO cl_abap_datadescr,
        lt_comp          TYPE abap_component_tab,
        ls_comp          TYPE abap_componentdescr,
        lr_wa            TYPE REF TO data,
        lr_tab           TYPE REF TO data.

DATA : lv_idx TYPE n LENGTH 2.

FIELD-SYMBOLS: <fs_field> TYPE ANY.
FIELD-SYMBOLS: <fs_wa>    TYPE ANY.
FIELD-SYMBOLS: <fs_tab>   TYPE table.

PARAMETER p_cnt TYPE i.

START-OF-SELECTION.

  DO p_cnt TIMES.
    lv_idx  = lv_idx  + 1.
    CONCATENATE 'column'  lv_idx  INTO ls_comp-name
    SEPARATED BY '_'.

    CALL METHOD cl_abap_elemdescr=>get_i
      RECEIVING
      p_result = ls_comp-type.

    INSERT ls_comp INTO TABLE lt_comp.
  ENDDO.

  CALL METHOD cl_abap_structdescr=>create
    EXPORTING
      p_components = lt_comp
    RECEIVING
      p_result     = lr_structdescr.

  CREATE DATA lr_wa TYPE HANDLE lr_structdescr.
  ASSIGN lr_wa->* TO <fs_wa>.

    lr_datadescr = lr_structdescr.


    CALL METHOD cl_abap_tabledescr=>create
      EXPORTING
        p_line_type  =  lr_datadescr
      RECEIVING
        p_result     =  lr_tabledescr .

  CREATE DATA lr_tab TYPE HANDLE lr_tabledescr.
  ASSIGN lr_tab->* TO <fs_tab>.

  DO  p_cnt  TIMES.
    DO.
      ASSIGN COMPONENT sy-index OF STRUCTURE <fs_wa> TO <fs_field>.
      IF sy-subrc NE 0.         EXIT.       ENDIF.
      <fs_field> =  sy-index.
    ENDDO.
    APPEND <fs_wa> TO <fs_tab>.
  ENDDO.

  LOOP AT <fs_tab> INTO <fs_wa>.
    DO.
      ASSIGN COMPONENT sy-index OF STRUCTURE <fs_wa>
           TO <fs_field>.

      IF sy-subrc IS NOT INITIAL. EXIT. ENDIF.
      WRITE  <fs_field>.

    ENDDO.

    WRITE  / .
  ENDLOOP.
```

[결과 3]



프로그램 Z_DYNAMIC_003

```
P_CNT                          5
```



프로그램 Z_DYNAMIC_003

```
프로그램 Z_DYNAMIC_003                                        1
-------------------------------------------------------------
          1         2         3         4         5
          1         2         3         4         5
          1         2         3         4         5
          1         2         3         4         5
          1         2         3         4         5
```