2001 추계학술발표회 논문집
한국원자력학회

MELCOR 코드의 구조변경을 위한 대상변수 자동검색 및 위치추적 프로그램 (MELvar) 개발

Development of a Target Variable Information Search Program for Structure Conversion in MELCOR Code

송용만, 박선희, 김동하
한국원자력연구소
대전광역시 유성구 덕진동 150

요 약

데이타 전달 및 저장을 위해 포인터 변수를 사용하는 MELCOR 코드의 데이타 체계를 코드구조 개편을 통해 모듈화된 체계로 바꾸면 코드개발자 및 사용자의 코드내부에 대한 이해를 크게 향상시킬 수 있다. 이러한 코드구조개편 작업을 자동적으로 수행하기 위해 코드변환 프로그램인 MELtoMID가 Fortran90 언어를 이용하여 개발된 바 있으며 여기서는 변경대상 변수를 사용자가 입력으로 처리하고 있다. 그러나 MELCOR 코드의 포인터 변수를 추적하면 변경대상 변수명 및 위치와 같은 정보를 추출할 수 있으며 이를 위하여 변경대상변수 자동검색 및 위치추적 프로그램 (MELvar)이 개발되었다. 본 프로그램은 MELtoMID 프로그램의 입력을 자동생성할 뿐 아니라 참조코드가 개선되어 새로운 판이 출시되어도 동일한 작업을 용이하게 반복할 수 있는 장점을 가진다. 본 프로그램은 개편대상인 12개 패키지의 수작업 결과와 대조하여 결과가 정확히 일치하는 것이 확인되었으며 MELCOR 코드의 국산화 과제에 계속 활용될 예정이다.

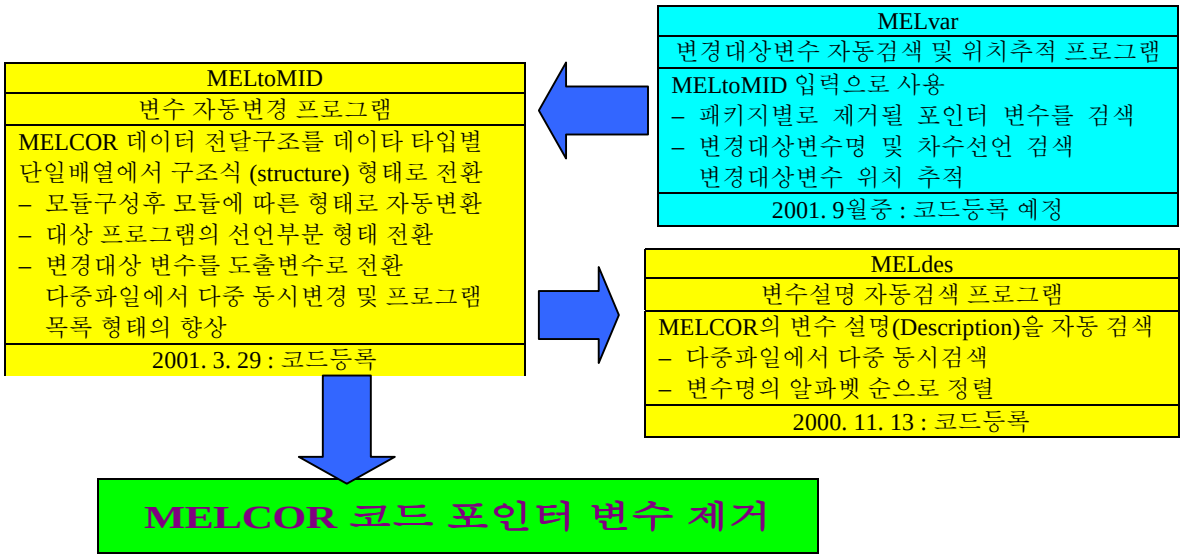
Abstract

MELCOR code restructuring is essential for code developers and users to easily understand code interior via substituting pointer variables used for data transfer and storage with a modularized data system. To perform a code restructuring process automatically, a code conversion program, MELtoMID, had been developed using Fortran90 language features and users should input target variables for conversion. But target variable information like a name and a position can be tracked from pointer variables in the MELCOR source list. Therefore, a Target Variable Information Search Program (MELvar) is being developed for this purpose. This program not only automatically generates input for MELtoMID program but also provides merit that the same conversion process can be easily repeated in case of a new version release. MELvar has been applied to 12 MELCOR packages and its results are confirmed from checking with manually searched results. MELvar is being used continuously in the domestication project of MELCOR code.

1. 서론

MELCOR[1] 중대사고 종합코드 데이터 전달방식의 단점을 보완하기 위해 FORTRAN90 동적메모리관리 (Dynamic Memory Allocation, DMM) 및 STRUCTURE 기법을 이용하여 변수 및 데이터 저장 방법을 개편하는 코드구조개편 작업[2]이 진행되고 있다. 기존의 데이터 전달방식

단점은 직접변수 대신 포인터(Pointer) 변수를 사용한 데서 기인한다. 포인터 변수의 사용은 컴퓨터 언어가 동적메모리 관리를 허용하지 못했던 예전에 메모리의 낭비를 없애기 위해 즐겨 사용하던 방식이다. 그러나 위 방식은 직접변수를 사용하지 못하기에 프로그램 내부에서의 변수추적이 아주 어려운 단점이 있다. 한편 최신 프로그램 언어인 FORTRAN90은 동적메모리 관리를 허용하므로 직접변수를 사용해도 메모리의 낭비를 방지할 수 있다. 또한 모듈을 이용한 STRUCTURE 기법을 사용하면 직접변수명에 수직포함관계 (hierarchy structure)를 표시 (수직포함관계가 표시된 직접변수를 이후 "도출변수"로 명명) 할 수 있어 변수명만으로 변수에 대한 이해를 크게 높일 수 있다. 따라서 가변적인 입력을 처리하기 위해 사용되었던 포인터 변수를 삭제하고 도출변수로 변경하는 구조개편 작업이 완료되면 MELCOR 코드내 모델의 개선 및 추가가 용이해지고 사용자의 편의성 (접근성 및 이해도)이 크게 향상되는 효과가 기대된다.



<그림.1> 코드구조개편 자동화 절차

이러한 변수 변경작업은 변경대상 프로그램의 길이 (40만 줄 이상)를 고려할 때 수작업에는 많은 시간과 노력이 필요하다. 따라서 그림.1에서 알수 있듯이 수작업을 최소화하기 위하여 변경대상 변수를 도출변수로 자동으로 바꾸어 주는 역할을 하는 자동변경 프로그램 (MELtoMID 1.0 [3])을 이미 개발하였다. 기 개발된 MELtoMID 프로그램은 변경대상 변수 및 도출변수의 목록을 입력으로 요구한다. 여기서 변경대상 변수는 포인터 변수가 호출 (call)되는 과정에서 사용되는 변수 (변수명이 대부분은 다르다)로 수작업으로 검색하기는 매우 어렵다. 특히 호출된 부프로그램에서 Dimension이 부여되는 과정도 추적해야 하므로 이를 자동화할 필요가 있다. 이를 위해 변경대상변수 자동검색 및 위치추적 프로그램 (MELvar)이 개발되었다.

2. MELvar 프로그램의 기능.구성 및 입.출력 양식

본 자동검색 및 위치추적 프로그램은 MELvar (**MEL**COR target **var**iable name and position search program)로 명명되었고 현재버젼은 1.0 이며 사용언어는 Fortran 90 이다. 본 프로그램의 주요 기능은 다음과 같다:

1. MELCOR 코드에서 패키지별로 제거될 포인터 변수를 검색한다.
2. 변경대상변수의 이름 및 위치를 모두 검색한다.
3. 모든 포인터 변수 각각에 대해 검색된 변경대상변수중 상이한 변수명(Dimension 포함) 및 검색회수 그리고 검색된 부프로그램명/위치를 정리된 형태로 출력한다.

본 프로그램은 다음 9개의 부 프로그램으로 구성되어 있으며 각각의 기능은 아래와 같다:

- MELvar.f90 : 전체 입.출력을 담당하는 주 모듈
- Pfinder.f90 : 제거대상 포인터 변수를 검색하는 모듈
- Mfinder.f90 : 변경대상 변수 및 위치를 추적하는 모듈
- Entry.f90 : 작업파일(들)에서 Entry문 검색후 중간파일(*.ent)에 출력
- Entry1.f90 : 중간파일(*.ent)에서 호출된 Entry문 검색
- Dcheck.f90 : 호출된 일반 부프로그램에서 해당 아규먼트의 차수선언 검색 모듈
- Dcheck1.f90 : 호출된 Entry 부프로그램에서 해당 아규먼트의 차수선언 검색 모듈
- Rblank.f90 : 작업 문단에서 연속된 공백을 제거하는 모듈
- Rsame.f90 : 검색된 변경대상변수중 동일한 이름을 처리하는 모듈

본 프로그램을 수행하면 초기입력으로 다음의 3가지를 요구한다.

1. Enter name of a PACKAGE to be worked :
2. Enter name of file(s) (can include wildcard [* and/or ?]) :
3. Do you want to check dimension declarations in a *.tmp file ? [y/n]

1번 항목은 작업대상 패키지명을 묻는 것이다. MELCOR 코드의 패키지는 24개이며 기존 패키지명이 아닌 이름을 입력하면 에러 메시지 (ERROR **** Invalid PACKAGE name was entered!!!')가 뜨게 된다. 2번 항목은 작업대상 파일명이다. 즉, 본 프로그램의 입력파일로서 F77toF90 프로그램의 출력이다. 여기서, F77toF90 프로그램은 fortran77로 쓰여진 기존 코드의 내용을 fortran90 언어로 변환하고 프로그램 목록 (Source List)의 형태를 사용자가 보기 좋은 형태로 바꾸는 역할을 하는 상업용 프로그램이다 [4]. 따라서 작업대상의 부프로그램을 F77toF90 프로그램을 이용하여 fortran90 특성을 가진, 사용자가 보기 좋은 형태로 바꾼후, 본 프로그램을 이용할 것을 추천한다. 입력파일명에 해당하는 파일이 본 프로그램의 수행 디렉토리내에 존재하지 않는 경우에는 에러 메시지 ('ERROR ****

Invalid file name was entered!!!')가 뜨게 된다. 3번항목은 변경대상변수의 Dimension 선언부분을 사용자가 확인하기를 원하는 경우를 대비한 선택사항이다. 본 선택사항에서 체크확인 옵션 (Y(y))을 선택하면 변경대상변수의 Dimension 선언부분이 중간출력파일 (*.tmp)에 표시되어 쉽게 확인이 가능하다.

출력파일은 중간출력 파일 2개와 정식출력 파일 1개로 이루어져 있다. 정식출력에서는 작업대상 패키지에 대해 검색된 모든 포인터가 첫 행에 표시된다. 두번째 행에는 각 포인터에 대해 검색된 변경대상변수가 배열의 크기와 함께 표시된다. 동일 포인터에 대해 검색된 변경대상변수는 다수가 존재하며 이중 동일한 이름을 가진 변경대상변수는 개수를 출력하여 표시된다. 세번째 행에는 검색된 변경대상변수가 위치한 부프로그램명이 표시되며 해당 부프로그램내 상세 위치는 중간출력인 *.tmp 파일에 표시된다. 중간출력 파일인 *.tmp 파일에는 포인터가 검색된 위치, 포인터가 호출문의 아규먼트로 나타나는 위치, 포인터가 재정의되는 위치, 호출되는 부프로그램내에 해당 아규먼트 및 차수가 선언되는 위치등과 같은 전반적인 위치정보가 표시된다. 한편 Entry 문이 나타날 경우 소속된 부프로그램의 Dimension을 검색해야 하므로 위 정보를 중간출력 파일인 *.ent 파일에서 따로 관리한다. 표.1에는 MP 패키지에 대한 MELvar의 정식 출력이 예시되어 있다.

3. MELvar 프로그램의 알고리즘 흐름도

MELvar 에서의 자동검색 및 위치추적 논리 흐름도는 그림.2에 정의되어 있고 각 과정에 대한 설명은 아래와 같다:

1. 사용자 입력으로 패키지의 이름(2-4글자, ‘xxx’로 일단 명명)을 읽어들인다.
2. xxxPRS.f 파일에서 모든 naming common (“<앞부분은 blank>COMMON /***DB/”를 제외)을 검색하고 여기에 선언된 포인터변수를 읽어들인다.
3. 검색된 naming common이 포함된 부프로그램에서 읽어들인 포인터변수를 포함한(argument 또는 argument의 array(size) 형태) 모든 호출문을 검색한다
4. 호출된 부프로그램에서 호출한 포인터변수 위치의 변수 (argument)명 (콤마의 개수 활용, 2차원 이상의 변수 존재 괄호쌍 인식 부가 이용)을 검색한다.
5. 호출된 부프로그램에서 해당 변수의 차수선언 (Dimension Declaration)이 존재하면 배열첨자를 포함한 변수명을 검색한다.
6. 검색된 변수명 (배열첨자 포함 또는 불포함)을 해당 포인터 변수명 및 호출된 부프로그램명과 함께 출력한다.
7. 한편, 입력된 포인터 변수가 연산등을 통해 호출문 이전 위치에서 재정의 되는 경우가 있는데, 이 경우에는 재정의된 변수명을 (4)번 항목에서 이용해야 한다.

8. 중간파일 (*.tmp 및 *.ent)에 다음을 부가로 출력한다.
   - xxxPRS.f9 파일내 포인터가 검색된 위치
   - 포인터가 호출문의 아규먼트로 나타나는 위치
   - 포인터가 재정의되는 위치
   - 호출되는 부프로그램내에 해당 아규먼트 및 차수가 선언되는 위치
   - Entry 문이 나타나는 부프로그램의 선언부분

4. 결과

개편대상인 24개 패키지중 15개 패키지에 대해 적용하고 결과를 확인하였다. 15개 패키지중 12개 패키지는 수작업 결과와 대조했으며 결과가 정확히 일치하는 것이 확인되었다. 이러한 결과중 SPR 패키지의 결과를 이용해 구성된 모듈을 표.2에 예시하였으며 위 모듈을 이용한 결과검증은 참고문헌 [4]를 참조하기 바란다. 계산시간은 규모가 큰 CVH/HS/COR 패키지에서 Pentium3/4 (300-1000 MHz) 개인용 컴퓨터를 사용하여 최대 10여분 정도가 소요되었다.

감사의 글

본 연구는 과학기술부의 원자력연구개발 사업의 일환으로 수행되었습니다.

참고문헌

[1] SNL (1990), MELCOR Computer Code Manuals, NUREG/CR -6119, SAND97-2398.

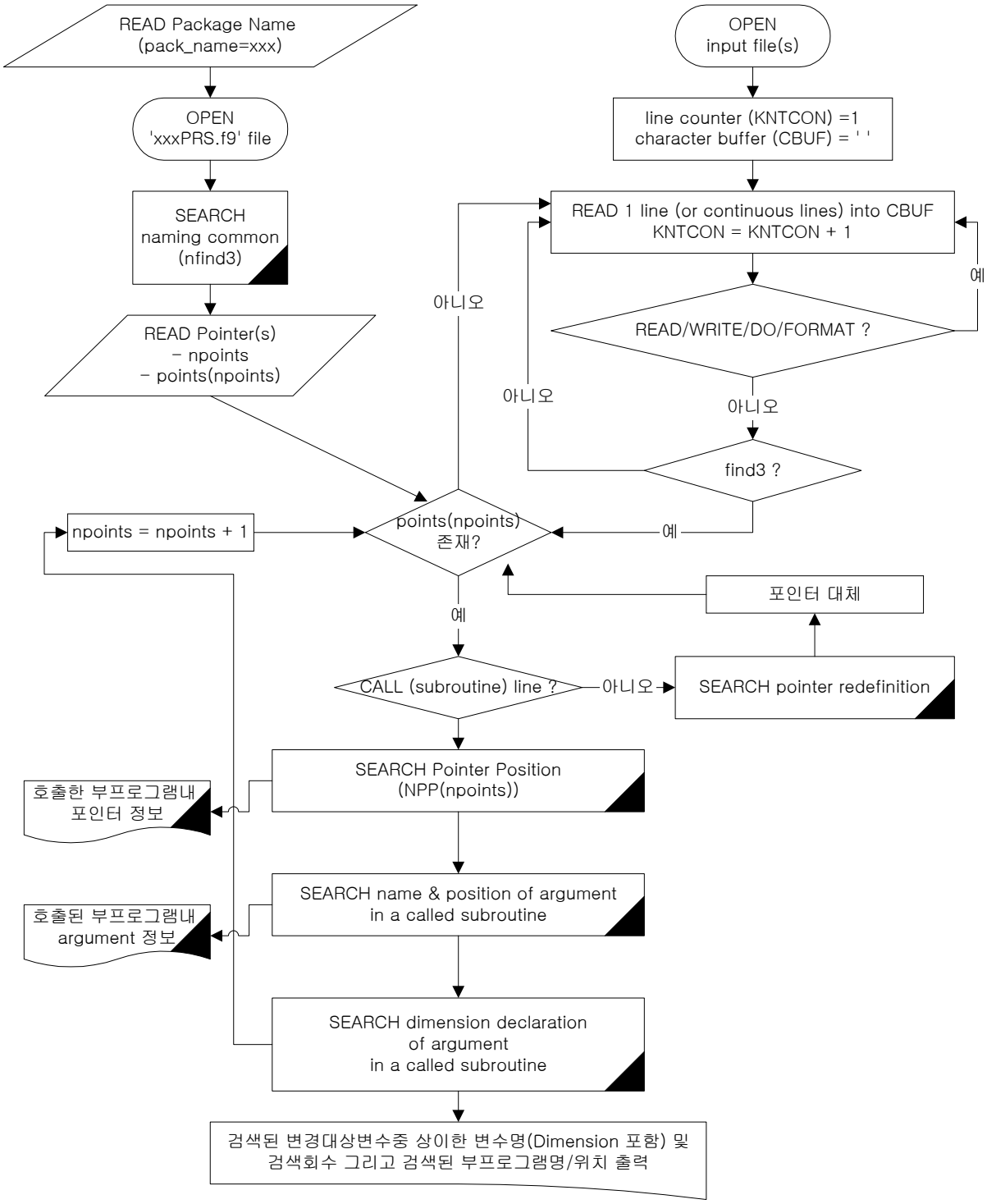
[2] 박선희외2 (2000), A Restructuring Proposal for MIDAS, Proceedings of the KNS ’2000 Spring Meeting.

[3] 송용만외2 (2000), Development of a Computer Program (MELtoMID) for Automatic Variable Conversion in MELCOR Code, Proceedings of the KNS ’2000 Autumn Meeting.

[4] KAERI (2000), 중대사고해석 코드 개발을 위한 MELCOR 개편시안, KAERI/TR–1536/2000.

<표.1> MP 패키지의 MELvar 작업 결과

MP Package의 개편대상 변수		1행	2행		3행
구성될 module의 구조		포인터	변경 전	변경 후	Subroutine
MP_NAM(NMATLS)	MATNAM	MMATNM	MATNAM(NMATLS) MATNMS(NMATS) ch*24	MP_NAM(NMATLS)%MATNAM	mpconr, mpps3, mprun, mpedt, mpchk, mpps2, mpmtcm, mpmtnb, mpprcm, mpprpm
MP_TBL(NPRMTL)	ITBPRM	MTBPRM	ITBPRM(NPRMTL)	MP_TBL(NPRMTL)%ITBPRM	mpps3, mprun, mpedt, mpchk, mpps2, mpevan, mpprcm, mpprpm
MP_CON(NPRCON)	XCNPRM	MTBCON	XCNPRM(NPRCON) XCNPRM(NPROPC,NMATLS)	MP_CON(NPRCON)%XCNPRM	mpedt, mpps2, mpevan, mpconr



<그림.2> 자동검색 및 위치추적 알고리즘 흐름도 (1개 파일 기준)

<표.2> SPR 패키지의 모듈 구성 예 (1/2)

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!    SPR (spray) package data     !!!!!!!!!!!!!!!!!!!!
!     IMPLICIT NONE
      MODULE SPR_MDL
!
!           *****  SPRAY GLOBAL DATA   *****
!                (SPRAY JUNCTION DATA)
       INTEGER :: NDSPJN,  NSPJUN
       REAL(8), ALLOCATABLE :: FRSPTR(:,:)
!                (SPRAY SOURCE DATA)
       INTEGER :: NDSPSR, NSPSRC, MXSPSZ, MXSPJN
!                (SPRAY VOLUME DATA)
       INTEGER :: NDSPVL, NSPVOL
!                (SUMP DATA)
       INTEGER :: NDSPSM, NSPVSM, NDSUMP, NSUMPS
!
!           *****  SPRAY JUNCTION DATA   *****
       TYPE SPR_J  ; SEQUENCE
          INTEGER :: ISPJNM,  KCVFM, KCVTO
          REAL(8) :: FRSPTI
       END TYPE
       TYPE (SPR_J), ALLOCATABLE :: SPR_JN(:)
!
!           *****  SPRAY SOURCE DATA   *****
            TYPE  SPR_S1 ; SEQUENCE
              REAL ::  DIAMO, DRFREQ, FLORTO, FLORTA, DRPIMS
            END TYPE
            TYPE  SPR_S2 ; SEQUENCE
              INTEGER  :: IVOLFR, IVOLTO
            END TYPE
       TYPE SPR_S ; SEQUENCE
         INTEGER :: ISPNUM
         CHARACTER(LEN=16)  :: SPNAME
         INTEGER  :: IVOL,   ISPCON
         LOGICAL  :: SPONOF
         INTEGER  :: ISRCVL
         REAL(8)  :: FALLHS, TDROPO
         INTEGER  :: ITMPCF
         REAL(8)  :: SPFLO,  SPFLOA
         INTEGER  :: IFLOCF
         REAL(8)  :: VWFREQ
         INTEGER  :: NSRCJN, NSRCSZ
         TYPE(SPR_S1), POINTER :: SR_SZ(:)
         INTEGER  :: IFDRY
         REAL(8)  :: ELDRY, ELWET
         LOGICAL  :: LDRYO_EV, LDRYN_EV, LDRYO_OD, LDRYN_OD
         INTEGER  :: ICVLSR, ICVXSR
         REAL(8)  :: HEXCHO_EV, HEXCHN_EV, HEXCHO_OD, HEXCHN_OD
         TYPE(SPR_S2), POINTER :: JNC(:)
       END TYPE
       TYPE (SPR_S), ALLOCATABLE :: SPR_SR(:)
!
```

<표.2> SPR 패키지의 모듈 구성 예 (2/2)

```
!           *****  SPRAY VOLUME DATA   *****
      TYPE SPR_V ; SEQUENCE
        INTEGER :: ISPCVL, ITOTVL, ISPSMP
        REAL(8) :: FALLHT, CAROVR, SPHEAT, SPMASS
      END TYPE
      TYPE (SPR_V), ALLOCATABLE :: SPR_VL(:)
!
!           *****  SUMP DATA   *****
            TYPE SPR_S3 ; SEQUENCE
              INTEGER  :: ICVSUM, ICVXSM
              REAL(8)  :: SUMPHT, SUMPMS
            END TYPE
            TYPE SPR_S4 ; SEQUENCE
              INTEGER :: ICVISM
            END TYPE
      TYPE (SPR_S3), ALLOCATABLE :: SPR_SM(:)
      TYPE (SPR_S4), ALLOCATABLE :: SPR_SM_CV(:)
!
!           *****  SCRATCH VARIABLE   *****
            TYPE SPR_S5 ; SEQUENCE
              REAL(8) :: TDFLO, DIAMF, TDFIN
            END TYPE
      TYPE SPR_S6 ; SEQUENCE
         REAL(8) :: TDINIT,  DIAMI
         TYPE (SPR_S5), POINTER :: VL(:)
      END TYPE
      TYPE (SPR_S6), ALLOCATABLE :: SC_SPR(:)
    END MODULE SPR_MDL
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```