2000 추계학술발표회 논문집
한국원자력학회

MELCOR 코드의 자동변수변경을 위한 프로그램 개발

Development of a Computer Program for Automatic Variable 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 MELtoMID를 Fortran90 언어를 이용하여 개발하였다. 본 프로그램은 기존 코드의 선언 부분(dimension, character, integer, real, 그리고 logical)에서 모듈 관련 변수를 제거하며, 현재 사용중인 변경대상 변수를 구성된 모듈에 의거하여 도출변수로 변환하고, 변경후 프로그램 목록의 모습을 사용자가 읽기 쉬운 형태로 유지시키는 기능을 보유하고 있다. 본 프로그램은 MELCOR의 일부 부프로그램을 대상으로 개편이전과 개편이후의 결과를 수작업으로 비교하여 동일함을 확인하였고, 50여개의 SPR 패키지 프로그램의 자동변환에 사용되었다. 본 프로그램은 참조코드가 개선되어 새로운 판이 출시되어도 동일한 작업을 용이하게 반복할 수 있는 장점을 가지며 MELCOR 코드의 국산화 과제에 활용되고 있다.

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 modularized data system. To perform a code restructuring process automatically, a code conversion program, MELtoMID, has been developed using Fortran90 language features. This program has three functions; to remove module-related variables in declaring parts like dimension, character, integer, real and logical declaration statements, to substitute conversion target variables used with derived (type) variables based on structured modules, and to keep up a good-looking format on the source list. This program is verified for several MELCOR subroutines by comparing before-conversion programs with those of after-conversion, through line-by-line proof reading and is extensively used for automatic conversion of over 50 subroutines in the SPR package. This program provides a merit that the same conversion process can be easily repeated in case of a new version release and is being used in the domestication project of MELCOR code.

1. 서론

중대사고 해석용 종합전산코드인 MELCOR 코드[1]를 제대로 파악하기 위해서는 전체 데이터의 저장 구조 및 저장 방법을 정확하게 알아야 한다. 현재의 MELCOR 코드는 데이터 전달 및 저장을 위해 네

개의 데이터 타입별 포인터를 사용하고 있다. 각각의 데이터 타입에는 최고 50만개의 저장 장소가 예약되어 있고, 그 안에서 사용자의 입력에 따라 저장 용량의 크기를 최적화하여 데이터를 저장한다. 이때 사용하는 데이터 저장방법은 팩키지별 데이터를 차례로 축적하는 방법으로, 임의의 팩키지에서 사용하고 있는 변수를 이용하고자 할 경우 실제로 쓰이는 변수 이름 대신 위치를 나타내는 포인터를 이용하여야 하며, 또한 포인터 위치를 파악하기 위해서는 전체 데이터의 저장 구조 및 저장 방법을 정확하게 이해하고 있어야 한다. 비록 MELCOR의 각 팩키지 안에서는 내용이 명확한 물리적 변수를 사용하고 있지만 팩키지 밖에서의 데이터 전달을 위해서는 포인터를 사용하고 있으므로 코드내부에 대한 전반적인 이해가 어려우며, 특히 새로운 모델의 추가시 연계작업이 복잡하여 많은 어려움이 따르는 실정이다.

따라서 기존의 MELCOR 데이터 저장 및 전달 체계를 모듈화시켜 코드 개발자나 사용자가 쉽게 코드 내부를 이해할 수 있도록 코드의 데이타구조를 개편할 필요가 있으며 이는 Fortran90 언어와 그 특성을 이용하여 MELCOR의 데이터 전달구조를 데이터 타입별 단일 배열에서 구조식 (structure) 형태로 전환하는 방안을 이용하면 된다[2]. 이러한 코드구조 개편과정에서 가장 중요한 부분은 데이터베이스 타입구조 (data base type structure)를 구축하는 것이다. 데이터베이스 타입구조는 코드의 유지/보수 능력 및 코드의 이해도 (readibility)를 크게 향상시키는데, 이러한 구조를 만들기 위해서는 우선 모듈 (modules)을 구성하고 이로부터 도출된 변수 (derived type variables)를 사용해야 한다. 이를 위해, 변경전 프로그램내 모든 패키지에서 도출변수를 결정한 후, 변경대상 변수는 도출변수 형태 (derived type data)로 변경되어야 한다[3]. 이러한 변경변수 작업은 변경대상 프로그램의 길이 (40만 줄 이상) 및 변경변수의 개수 (SPR 패키지만 56개)를 고려할 때 수작업에는 많은 시간과 노력이 필요하다. 특히, 프로그램 선언부분의 처리 및 일부 이차원 변수 첨자 (index)의 역전등에 따른 작업실수를 고려할 때 수작업후의 확인절차에 이중적인 노력이 필요하게 된다. 따라서 수작업을 최소화하기 위하여 변경전 프로그램내 변경대상 변수를 도출변수로 자동으로 바꾸어 주는 역할을 하는 자동변경 프로그램을 개발하였다. 본 프로그램을 이용하면 참조코드가 개선되어 판(version)이 바뀌어도 동일한 작업을 용이하게 반복할 수 있는 장점이 있다. 또한 개발 프로그램을 이용하여 코드 전체의 구조를 체계적으로 변환하게 되면 앞으로 예상되는 기존 모델의 개선이나 추후 개발되는 새로운 모델의 접목시 어려움 없이 코드 내부를 이해할 수 있어 용이하게 코드 국산화 작업을 수행할 수 있는 장점을 갖게 된다. 본 자동변경 프로그램은 MELtoMID (MELCOR to MIDAS)으로 명명되었고 사용언어는 Fortran90 이다.

본 프로그램의 주요 기능은 다음과 같다:

1. 변경전 프로그램의 선언부분 (DIMENSION/CHARACTER/INTEGER/REAL/LOGICAL)에서 모듈관련 변수 또는 첨자를 제거한다.
2. 변경전 프로그램에서 변경대상 변수를 구성된 모듈에 의거하여 도출변수로 변경한다.
   - 도출변수는 1차원 및 2차원 배열 (dimension array)을 모두 지원하며 추후 필요시 3-4 차원 배열 변수를 지원하도록 확장이 가능하다.
   - 2차원 이상의 배열에는 index 위치가 바뀐 경우도 처리 가능하다.
- 다수 파일의 동시처리 및 파일명에 wildcard (* 또는 ?)의 이용이 가능하고 대.소문자 변수명의 구분이 가능하다.
3. 변경후 프로그램 목록 (Source List)의 형태는 사용자가 보기 좋은 형태를 유지시킨다.

2. 입.출력

본 프로그램을 수행하면 초기입력으로 다음의 5가지를 요구한다.

1. Enter name of file(s) (can include wildcard [* and/or ?]) :
2. Enter new extension name of output file(s) [default=f90] :
3. Enter name of before-conversion variable(s) (need blank between variables)
   (Or just hit the "ENTER(CR)"key to use default file(=conOLD.in))
4. Enter name of after-conversion variable(s) (need blank between variables)
   (Or just hit the "ENTER(CR)"key to use default file(=conNEW.in))
5. Do you want a CHECK MESSAGE in case of change? [y/n]
   (CHECK MESSAGE = !! NOTICE: Modified by MELtoMID)

1번 항목은 본 프로그램의 입력파일명을 묻는 것이다. 본 프로그램의 입력파일은 Fortran90 프로그램 (또는 다음절의 프로그램의 입력관련 제한사항을 만족하는 Fortran 프로그램)이다. 입력파일명의 extension은 출력파일명의 extension 기본값 (*.f90)과의 혼동을 피하기 위하여 (*.f90)을 사용하지 말고 다른 extension (예: (*.f9))을 사용하기를 추천한다. 입력파일명에 해당하는 파일이 본 프로그램의 수행 디렉토리내에 존재하지 않는 경우에는 에러 메시지 ('ERROR **** Invalid file name was entered!!!')가 뜨게 된다.

2번 항목은 사용자가 원하는 출력파일명의 extension 이다. 출력파일명중 extension을 제외한 부분은 입력파일명과 동일하며 이는 사용자가 제어할 수 없다 (꼭 필요하면 윈도우상에서 재명명 하기 바란다). 출력파일명의 extension 기본값은 *.f90 이다.

3번 항목은 변경대상변수(before-conversion variables) 파일 또는 사용자가 원하는 변경대상변수 이름이다. 본 항목에서 엔터키만을 치면 변경대상변수 파일(conOLD.in)을 이용하게 되고, 아니면 변경대상변수 이름을 입력창에 직접 입력하면 된다. 변경대상변수 이름의 사이에는 최소 1칸의 공백이 있어야 하며 변경대상변수 이름 1개의 최대길이, 최대개수, 및 (전체)글자수는 일단 15자, 1000개, 및 8000자로 각각 제한하였다. 변경대상변수 파일명은 conOLD.in 으로 고정하였으며, 파일내에는 변경대상변수를 1-80행 사이에 최대 100줄까지 기록할 수 있고, 나머지 제한점은 입력창에 입력시와 동일하다.

4번 항목은 변경후변수(after-conversion variables) 파일 또는 사용자가 원하는 변경후변수 이름이다. 본 항목에서 엔터키만을 치면 변경후변수 파일(conNEW.in)을 이용하게 되고, 아니면 변경후변수 이름을 입력창에 직접 입력하면 된다. 변경후변수 이름의 사이에는 최소 1개의 공백이 있어야 하며 변경후변수 이름 1개의 최대길이, 최대개수, 및 (전체)글자수는 일단 15자, 1000개, 및 8000자로 각각 제한하였다 (입력완료시 'Enter'키). 변경후변수 파일명은 conNEW.in 으로 고정하였으며, 파일내에는 변경후변수를 1-80행 사이에 최대 100줄까지 기록할 수 있고, 나머지 제한점은 입력창에 입력시와 동일하다. 주의할 사항은 3번의 변경대상변수의 개수와 여기서의 변경후변수의 개수가 일치하지 않는 경우에는 에러 메시지 ('Error: Mismatch in number of before-/after-conversion variable(s)')가 뜨게 된다.

5번항목은 내용이 변경된 부분을 사용자가 확인하기를 원하는 경우를 대비한 선택사항이다. 본 선택사항에서 변경확인 옵션 (Y(y))을 선택하면 내용이 변경된 줄 (또는 내용이 변경된 연속줄의 첫줄)의 바로 윗줄에 다음의 메시지 ('!! NOTICE: Converted by MELtoMID')가 출력파일에 표시되어 쉽게 확인이 가능하다.

출력에서는 작업대상 변수가 도출변수 형태로 변경된다. MELCOR 코드의 SPR 패키지에 대해 작업대상인 변경대상 변수 및 도출변수의 일부 및 MELtoMID 프로그램의 입력 예를 아래의 <표.1> 및 <표.2>에 각각 예시하였다.

<표.1> SPR 패키지의 변경대상 변수 및 도출변수의 예

변경대상 변수	도출변수
ISPJNM (NDSPJN)	SPR_JN(NDSPJN)%ISPJNM
DIAMO (MXSPSZ, NDSPSR)	SPR_SR(NDSPSR)%SR_SZ(MXSPSZ)%DIAMO
IFDRY (NDSPSR)	SPR_SR(NDSPSR)%IFDRY
IVOLFR (MXSPJN, NDSPSR)	SPR_SR(NDSPSR)%JNC(MXSPJN)%IVOLFR

<표.2> MELtoMID 프로그램의 입력 예

conOLD.in	ISPJNM DIAMO IFDRY IVOLFR
conNEW.in	SPR_JN SPR_SR%SR_SZ SPR_SR SPR_SR%JNC

도출변수의 명명은 변경대상 변수의 차원에 따른 수직구조 (hierarchy structure)의 형태와 관련되어 있으며 자세한 내용은 참조문헌 [3]에 언급되어 있다. 단, 수직구조 이름과 '%'로 연결된 최종부분의 이름은 명명방식을 아직 결정하지 않은 관계로 변경이전의 이름을 그대로 사용하고 있다. 한편, 부프로그램을 call 하는 구조를 사용자가 확인하기를 원하는 경우에는 프로그램 수행후 call_nest.txt 파일의 내용을 확인하면 된다. 위 파일에는 변경이 수행된 부프로그램의 이름과 각 부프로그램마다 call 하는 다른 부프로그램의 이름이 표시되어 call의 포함구조 (nesting structure)를 확인하는데 도움을 준다.

3. 자동전환 알고리즘 흐름도

<그림.1>에 보인 것처럼 자동전환 프로그램의 논리 흐름도는 아래와 같다:

1. Fortran90 프로그램 또는 Fortran90으로 변환된 파일 (또는 파일들) 및 변경대상/ 변경후 변수를 입력파일로 연다.
2. 읽어 들인 연속줄의 개수 (continuation line counter : KNTCON) 및 문자 버퍼(character buffer :CBUF)를 초기화 한다.
3. 입력파일에서 각 줄의 73행까지 읽어 들인다 (단, 연속줄이 있는 경우는 연속줄을 모두 읽어 들인다).
4. comment 줄의 여부를 확인하고 맞으면 출력파일에 변경없이 적는다.
5. 'FORMAT' 줄의 여부를 확인하고 맞으면 출력파일에 변경없이 적는다.
6. 'CALL subroutine' 줄의 여부를 확인하고 맞으면 출력파일에 변경없이 적는다.
7. 선언문 (DIMENSION/INTEGER/REAL/CHARACTER/LOGICAL)의 여부를 확인하고 맞으면 대상변수를 검색하여 아래와 같은 방식으로 삭제한다.

- (CHARACTER/LOGICAL) 경우
  - 변수가 없거나 검색된 변경변수가 무차원인 경우 재검색
  - 1-D(일차원 변수) 이상인 경우 첨자만 삭제
- (DIMENSION/INTEGER/REAL) 경우
  - 변수가 없는 경우 재검색
  - 검색된 변경변수가 무차원인 경우 해당변수 삭제
    - 해당 변경변수만 존재 : 출력파일에 적지 않고 (즉, WRITE 없이) 빠져나간다
    - 처음에 위치 : 삭제 (해당 변경변수+뒤쪽의 콤마)
    - 나머지 경우 : 삭제 (콤마+해당 변경변수)
  - 1-D(일차원 변수) 이상인 경우 첨자포함해 삭제

- 해당 변경변수만 존재 : 출력파일에 적지 않고 (즉, WRITE 없이) 빠져나간다
- 처음에 위치 : 삭제 (해당 변경변수+뒤쪽의 콤마)
- 나머지 경우 : 삭제 (콤마+해당 변경변수)

8. 선언문이 아닌 경우, 대상변수를 검색하여 아래와 같은 방식으로 도출변수로 변경한다
   - 변수가 없거나, 검색된 변수가 0-D(무차원 변수) 경우 재검색
   - 1-D(일차원 변수) 경우 변경
   - 2-D(이차원 변수) 경우 변경
   - 3-D(삼차원 변수) 및 4-D(사차원 변수) 경우, 에러 메시지
9. 줄을 연속기호('&')를 사용해 분리한후, 출력파일에 적는다

4. 프로그램의 특성, 제한점 및 유의사항

본 프로그램의 주요 특성 및 제한/유의 사항을 다음에 정리하였다.

- Fortran90 프로그램에서 행(column)은 최대 72행으로 제한되고 73행은 연속줄기호(&)의 표시로 이용되므로 입력에서 허용되는 행의 개수는 최대 73행 이다. 만약 연속줄기호가 73행 이전에 위치하면 연속줄기호 이후에는 73행 까지 빈칸(blank) 이어야 한다. 입력(프로그램)에서 연속줄 표시를 이용한 연속줄의 최대 허용개수는 200 이고, 연속해서 한번에 읽어들일 수 있는 최대 글자수는 (200*73-1) 이다.
- 내용이 변경된 부분을 사용자가 확인하기를 원하는 경우에는 초기 입력창에서 변경확인 옵션을 선택하면 된다. 단, 'IF (조건) CALL문'의 형태에서 call문의 매개변수가 변경변수에 해당하는 경우, 현재는 프로그램 내부에서 중간에 변경되었다가 원상태로 회복시키기 때문에 '!! NOTICE: Converted by MELtoDIM' 메시지가 출력파일에 나타난다. 부 프로그램을 call 하는 구조를 사용자가 확인하기를 원하는 경우에는 프로그램 수행후 call_nest.txt 파일의 내용을 확인하면 된다.
- 도출변수명에서 수직구조의 표현에 따른 연장길이 (도출변수명의 길이 – 변경이전 길이)는 우선 최대 15글자로 정했으며 필요시 조정이 가능하다. 여기서, 변경이후 변수의 연장길이는 '%'를 포함한 순전히 늘어나는 길이로 정의한다. 변경 대상변수명은 대/소문자가 모두 가능하나 (변경대상 변수명이 소문자 이면 도출변수명도 소문자, 변경대상 변수명이 대문자 이면 도출변수명도 대문자로 바뀐다), 하나의 단어안에 대/소문자가 혼용되면 안된다. 따라서 현재의 MELCOR 프로그램 목록은 comment line 이외는 모두 대문자를 쓰고 있으나, 코드수정시에 각 단어는 모두 대문자 (또는 모두 소문자)만으로 작성해야 하며 대문자와 소문자를 섞어쓰면 안된다.

- 부 프로그램을 call 할때 매개변수가 변경대상 변수일 경우에는 현재는 변경하지 않는다. 이는 아직 변경원칙이 정해지지 않았기 때문이며 추후에는 본 경우에 대한 변경기능이 추가될 예정이다 (단, 함수(function)의 매개변수가 변경대상 변수일 경우에는 변경된다). 한편, call문이 문장의 처음이 아닌 중간에 나타나는 경우 (예: IF (조건) CALL문)에는 call문 이전에 변경대상 변수가 나타나면 변경된다.
- (DIMENSION/CHARACTER/INTEGER/REAL/LOGICAL) 선언의 경우, 변경변수명의 검색시 동일한 변수는 한번 읽은 부분에서 한번이상 나타나지 않는다. (DIMENSION/CHARACTER/INTEGER) 선언의 경우, 해당 변경변수의 삭제시 변경변수가 무차원이 아닌 경우는 index를 포함해서 삭제해야 하며, 변수의 위치 (처음, 중간, 끝)에 따라 삭제해야 하는 콤마의 위치가 다르므로 (예: 처음에 위치 : Delete (해당 변경변수+뒤쪽의 콤마), 나머지 경우 : Delete (콤마+해당 변경변수)) 주의해야 한다. 또한 삭제변수 홀로 존재하는 경우는 선언어 (declaration statement)를 포함한 줄 전체를 삭제해야 한다.
- (DIMENSION/CHARACTER/INTEGER/REAL/LOGICAL) 선언이 아닌 경우, 검색된 변경변수가 0-D (무차원 변수) 경우에는 local 변수이므로 변경하지 않고 1차원 이상의 경우에만 변경한다. 2차원 경우, 변경변수의 index는 일반적으로 위치가 역전되어 변경되나 (예 : DRPIMS (ISIZE, ISPATH) → SPR_SR(ISPATH)%SR_SZ(ISIZE)%DRPIMS) 일부변수의 경우 index의 위치가 역전되지 않는 경우도 있다. 변경시의 작업순서는 변경작업에 의해 늘어나는 SPACE를 chacarter buffer (CBUF)에서 먼저 늘려놓고 대상변수 뒷부분을 우선 옮긴뒤 작업해야 한다. 한편, 3차원 및 4차원 변수가 검색되는 경우에는 아직 변경기능이 없으므로 에러 메시지 ('ERROR : more than 2-D')가 표시되어 확인만이 가능하다.
- (DIMENSION/CHARACTER/INTEGER/REAL/LOGICAL) 선언의 경우에서의 대상변수 삭제 및 선언이 아닌 경우의 대상변수 변경을 위한 변경변수명을 검색할 때, 변수명의 포함관계 (예: IVOL과 IVOLFR)를 고려해 (변경변수명+Blank)를 검색해야 한다.
- 변경변수의 첨자(subscript)로서 또 다른 변경변수가 나타나는 경우, 그리고 변경변수의 첨자로서 (정수)연산식이 사용될 때 연산식 내에 또다른 변경변수가 나타나는 경우, 2회 이상의 변경이 필요하다. 이러한 경우들의 변수변경을 위해서는 첨자내의 괄호를 쌍(괄호열음/괄호닫음)으로 인식하는 논리[3]가 필요하다. 또한 배열에서 1차원 및 2차원의 구분을 위해 현재는 괄호안의 콤마(,)의 존재유무 및 위치를 이용하고 있으나, 변경변수의 첨자로서 또다른 2차원 변경변수가 나타나는 경우(예: A(B(1, 2), C))에는 추가적인 고려가 필요하다. 현재 변경변수의 첨자로서 2차원 배열의 변경변수가 이용되는 경우에 대한 임시적 대비는, 2차원 변경변수를 0/1-차원 변경변수보다 먼저 변경하는 방안 (즉, 입력파일의 변경변수 목록에서 2차원 변경변수가 0/1-차원 변경변수보다 먼저 나타나는

방안)을 이용하면 된다. 다중 포함구조의 경우도 가장 안쪽의 변경변수부터 변경하는 방안이 사용될 수 있으나 사용자가 불편하므로 필요 알고리즘을 추후 구현할 예정이다.

- 사용자가 보기좋은 형태를 유지하기 위해, 연속줄의 내용이 변경된 경우 첫줄이후 문장의 시작위치는 첫줄의 시작위치에 맞춘다 (단, 첫줄의 시작위치가 7행 미만일 경우 (즉, 문장번호등으로 인해), 시작위치를 7행으로 정한다). 연속된 빈칸은 1개의 빈칸을 제외하고 나머지는 없앤다 (단, 왼괄호 이후에는 1개의 빈칸이라도 없앤다). 그리고 줄을 연속기호를 사용해 분리할때, 콤마(',')/왼괄호('(')/오른괄호(')')/'%'를 제외한 단어의 중간에 끊기지 않게 조정한다.
- 연속줄 경우를 제외하고 연속줄기호 (&) 표시가 나타나는 경우(예 : write 문)가 있으므로 연속줄기호의 정의는 연속줄기호 이후의 빈칸 여부에 주목해야 한다. 한편, Fortran90 프로그램은 연속줄 기호를 73행에 고정시킨다. 본 프로그램은 '&'표시 이후 72행까지 빈칸이면 연속줄기호로 받아들이므로 코드수정시에 연속줄기호를 특별히 73행에 고정시킬 필요가 없다.
- 현재 변수변경시 연장되는 변수부분의 최대길이는 15자로 제한된다.

5. 결과 검증 및 추후계획

개편 시안 대상으로 선정한 MELCOR 코드의 SPR 팩키지에서 사용하고 있는 변수를 Fortran90의 특성을 활용할 수 있는 도출변수 형태로 바꾸어 모듈을 구성하였다. 이 모듈에 따라 자동변환 프로그램을 사용하여 몇 개의 부프로그램 (spryhm, sprrun, spryvl)을 변환하였으며 그 결과와 수작업 결과를 비교하여 이상이 없음을 확인하였다 (<그림.2> 참조). 이후 SPR 팩키지에 속해있는 약 50여 개의 전체 부프로그램을 변환하였으며 변환전.후의 계산결과가 정확히 일치하는 것이 확인되었다.

앞으로 포함구조를 가진 call 부프로그램의 매개변수 (argument)의 자동변경 기능과 3차원 이상의 행렬 변수에 대한 변환 기능을 추가한 MELtoMID 정식버젼을 출시하여 전체 프로그램 변환에 활용할 예정이다.

감사의 글

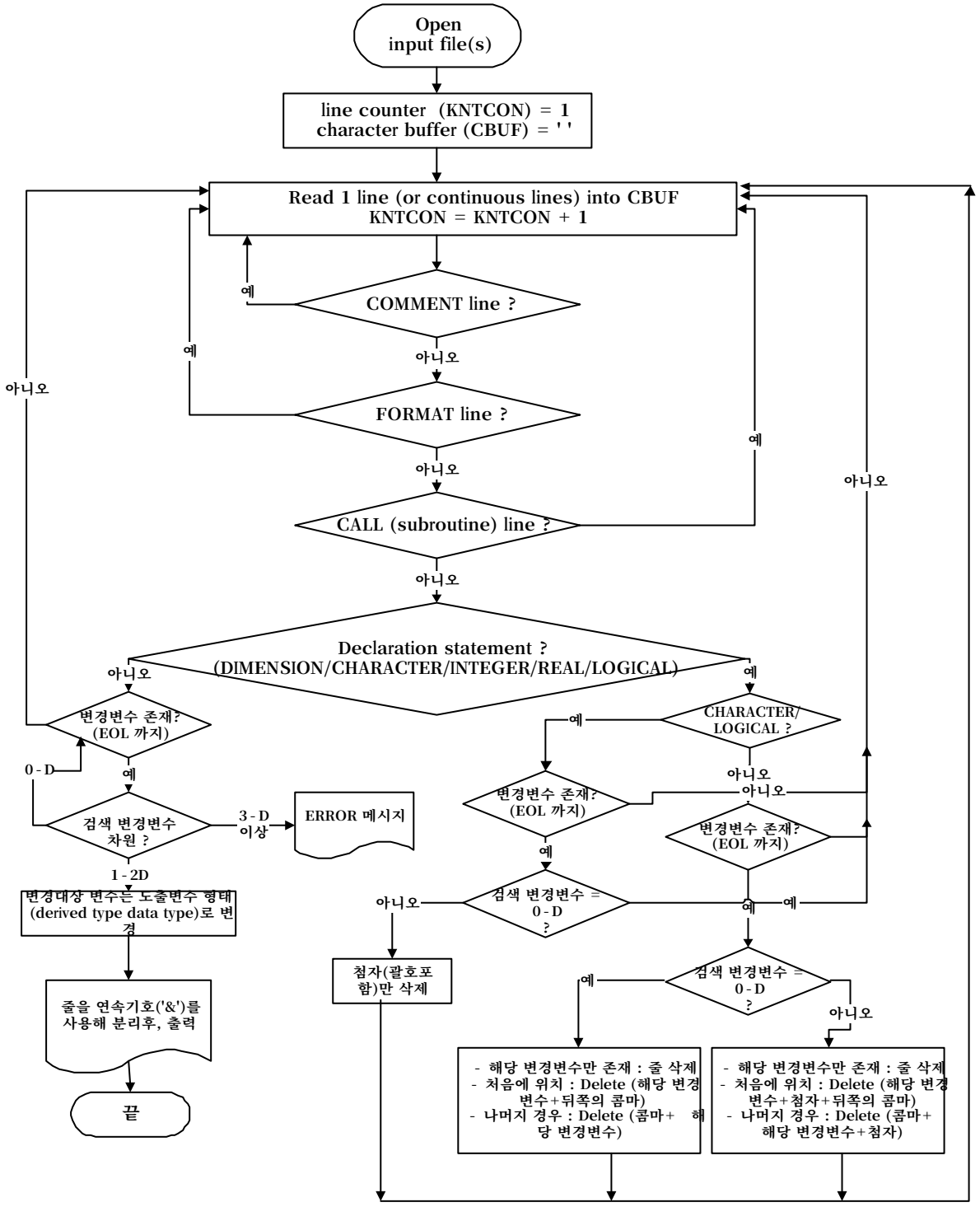
본 연구는 과학기술부의 원자력연구개발 사업의 일환으로 수행되었습니다.

참고문헌

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

[2] **박선희외**2 (2000), A Restructuring Proposal for MIDAS, 2000 Spring KNS Conference.

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



<그림.1> 자동전환 알고리즘 흐름도

```
SUBROUTINE TEST (ISPJNM, DIAMO, KCVFO)
!
!     DECLARE ARRAYS IN FORMAL PARAMETERS
!
      DIMENSION ISPATH (NDSPJN), KCVFO (NDSPJN), DIAMO (MXSPSZ, NDSPSR)
      INTEGER(4) ISPJNM (NDSPJN), IFDRY (NDSPSR)
      LOGICAL IVOLFR (MXSPJN, NDSPSR), LDRYO (NDSPSR), LDRYN (NDSPSR)

      IF (IVOLFR.GT.0) THEN
        DO 10 ISMP = 1, NSUMPS

          ISPJNM (NDSPJN) = ISPATH (NDSPJN) / KCVFO (NDSPJN) + 12345678 &
* IFDRY (NDSPSR)
          LDRYO (NDSPSR) = IVOLFR (MXSPJN, NDSPSR)
   10   END DO
      ENDIF

      WRITE (CHERRR, 610) ISPATH, DIAMO (MXSPSZ, NDSPSR)
  610 FORMAT('SPRAY SOURCE ',I5,' HAS ',I5)
!
      RETURN
      END SUBROUTINE TEST


SUBROUTINE TEST (ISPJNM, DIAMO, KCVFO)
!
!     DECLARE ARRAYS IN FORMAL PARAMETERS
!
      DIMENSION ISPATH (NDSPJN), KCVFO (NDSPJN)
      LOGICAL LDRYO (NDSPSR), LDRYN (NDSPSR)

      IF (IVOLFR.GT.0) THEN
        DO 10 ISMP = 1, NSUMPS

!! NOTICE: Converted by MELtoMID
          SPR_JN(NDSPJN)%ISPJNM = ISPATH (NDSPJN) / KCVFO (NDSPJN) +    &
          12345678 * SPR_SR(NDSPSR)%IFDRY
!! NOTICE: Converted by MELtoMID
          LDRYO (NDSPSR) = SPR_SR(NDSPSR)%JNC(MXSPJN)%IVOLFR
   10   END DO
      ENDIF

!! NOTICE: Converted by MELtoMID
      WRITE (CHERRR, 610) ISPATH, SPR_SR(NDSPSR)%SR_SZ(MXSPSZ)%DIAMO
  610 FORMAT('SPRAY SOURCE ',I5,' HAS ',I5)
!
      RETURN
      END SUBROUTINE TEST
```

<그림.2> 자동전환전.후의 프로그램 목록 비교