UIT Report Vol. 13, No.2, pp.359~366, 1982 연구논문집 제13권 제2호 pp.359~366, 1982

Ada 컴파일러의 구성에 대한 연구
—Ada 데이타 형식의 표현방법에 관해서—

우 치 수·박 양 수
전자계산학과
(1982.6.30 접수)

〈요 약〉

Ada 프로그래밍 언어에 대한 컴파일러의 구성시 컴파일러의 여러 단계 중에서 기억장소의 할당에 대한 문제, 특히 프로그램의 실행중 기억장소에 기억시켜야 한 데이타 형식의 표현방법, 그리고 변수의 표현방법에 대한 문제가 야기된다. Ada 언어의 데이타 형식 사용방법에 의하면 일부 데이타 종속형식에 대한 정보는 실행중에이 결정할 수 있게 되어 있는데, 이러한 종속형식 정보는 프로그램 실행중에 기억장소에 기억시켜 놓아야 하며, 이러한 데이타 형식의 변화는 결과적으로 변수에 대한 기억장소의 할당에도 영향을 미치게 된다. 본 논문에서는 이러한 Ada 언어의 데이타 형식에 대한 개념을 기억장소내에 효과적으로 표현하는 방법을 취급하였다.

A Study on Implementation of Ada Compiler
—Representation for Ada Types and Variables—

Woo, Chi Su · Park, Yang Su
Dept. of Computer Science
(Received June 30, 1982)

〈**Abstract**〉

When we are implementing an storage allocation phase during construction of an Ada compiler, the problem of representation for Ada types and variables is caused. The type and subtype facilities of the Ada programming language permit some subtype information to be determined dynamically. This subtype information requires a runtime representation, and its dynamic nature influences the representation of variables. This paper presents a representation for Ada types and variables.

I. 서 론

컴파일時 모든 변수들에 대한 성격(attribute)을 알 수 있다면, 이러한 변수들의 성격을 기초로 해서 컴파일러는 필요한 기계어를 생성할 수가 있다. 하지만 Ada 언어에 있어서 이러한 정보는 실행時에 동적으로 제공되는 경우가 있다. 예를들면 배열에서 상한값(upper bound)과 하한값(lower bound)을 컴파일時에 알 수 없는 경우가 있으며, 또한 實媒介變數의 데이타 형식과 假媒介變數의 데이타 형식이 일치하지 않는 경우도 있을 수 있다. 이러한 경우에 컴파일러는 필요한 데이타 값에 대한 기억장소를 할당할 수가 없으며, 따라서 효과적인 기계어를 생성할 수가 없게 된다.

이러한 문제를 해결하기 위해서 컴파일러는 데이타값을 갖는 변수에 기억장소를 할당함은 물론 데이타 형식에 대한 정보도 기억장소에 표현하여야 한

다. 이러한 데이타 형식 정보를 기억장소에 표현한 것을 데이타형식 기준표(data type template)라고 하며 실행중에 결정되는 데이타의 성격은 이곳에 기록되게 된다. 이러한 형식기준표의 내용은 실행중에 변경될 수도 있다(즉, 데이타의 성격이 변경될 수도 있다.).

어떤 변수에 대한 성격을 프로그램의 실행시에 알 수 있다면, 이러한 변수들에 대한 기억장소의 성격을 알 수 없으므로 컴파일러는 이러한 변수들에 대해서 기억장소를 할당할 수가 없다. 이런 경우, 컴파일러가 할 수 있는 일은 변수의 값이 기억될 데이타 영역에 일정한 크기의 형식기준표를 기억시켜 놓는 일이다. 그후 실제 프로그램 실행 중에 변수의 성격이 결정되면 기억장소 할당 프로그램에 의해서 해당 변수에 대한 기억장소가 할당되며, 이 기억장소의 주소가 형식기준표에 기록된다. 이렇게 해서 필요한 데이타의 값을 이 형식기준표를 통해서 찾아볼 수가 있다.

본 논문에서는 Ada 언어에 나타나는 각각의 데이타 형식에 대해서 효과적인 형식기준표의 작성방법을 연구하였다.

Ⅱ. Ada의 데이타 형식

프로그램의 실행시에 정수형 변수, 배열 변수, 레코드 변수와 같은 변수들을 기억장소에 기억시켜야 하는 동시에 데이타 형식에 대한 정보도 기억시켜야 한다. 일반적으로 프로그램의 실행시에 많은 양의 데이타 형식에 대한 정보가 제공되어야 한다. 예를들면, 대입문에서 범위검사(range checking)를 할 수 있도록 정수형 변수가 가질 수 있는 값의 범위를 알고 있어야 하며, 레코드 변수인 경우, 구별란(discriminant)의 값을 알고 있어야 구성요소(field)에 대한 착오를 검사할 수 있고 배열 변수라면, 구성원소에 대한 한계값과 데이타 형식 정보를 알아야 한다.

이처럼 특정 변수와 그 데이타 형식을 연결시켜 주는 이외에, 실행시 데이타 형식 자체를 표현할 수 있어야 한다. 예를들면 특정 데이타 형식의 성격(예를들면 배열 형식에서 하한 첨자값)을 알 수 있어야 하는데, 변수의 표현과 비교해서 이러한 데이타 형식을 형식 설명표(type descriptor)라고 한다.

Ada의 데이타 형식중 실행시 기억 장소의 할당에 영향을 미치는 요소를 정리하면 다음과 같다.

1. 제한요소와 종속형식

간략하게 말하면 Ada에는 데이타 형식(type)과 종속형식(subtype)이 있다. 종속형식이란 제한요소(constraints)를 포함하고 있는 데이타 형식을 의미한다. 예를들면, 두개의 변수가 같은 형식이면서 서로 다른 제한 요소를 가지고 있을 때 이 변수들은 서로 다른 종속형식이 된다. 제한요소는 실행시에 결정될 수도 있으며 일반적으로 어떤 연산자가 연산식의 종속형식의 일치를 요구한다면, 실행시에 종속형식의 일치성을 검사해야 한다.[1,2]

데이타 형식에 대한 제한요소는 몇가지 종류로 구별할 수 있다. 수치(scalar)형식에는 범위제한요소가 있을 수 있으며, 배열형식에는 배열의 한계(bound)를 결정해 주는 첨자제한요소가 있을 수 있는데, 배열에서 첨자를 사용하는 경우, 그리고 배열전체에 값을 대입하는 경우, 혹은 배열전체를 비교하는 경우, 이러한 배열의 한계는 반드시 검사되어야 한다. 레코드형식은 같은 레코드內에서 또 다른 레코드 구성요소(field)에서 배열한계로 사용되는 레코드 구성요소를 가질 수 있는데, 이러한 레코드 구성요소를 레코드결정인자라고 한다. 한 레코드의 레코드결정인자는 결정인자 제한요소(discriminant constraint)에 의해서 특정값들로 제한을 받을 수 있다. 마지막으로 악세스(access) 형식은 악세스되는 형식이 배열인 경우에는 첨자제한요소를, 그리고 레코드인 경우에는 결정인자 제한요소를 가질 수 있다.

2. 레코드內의 동적배열요소

대부분의 경우, 변수의 종속형식 제한요소는 컴파일러에 의한 기억장소 할당시 고정된다. 이러한 사실은 프로그램에서 새로운 block으로 들어가는 경우, 그리고 heap기억장소 할당시에도 해당된다. 그러나 배열한계(array bound)로서 사용되는 레코드결정인자를 가지고 있는 레코드 변수인 경우에는 예외이다. 다음 例에서 변수 *z*가 그러한 레코드이다.

```
subtype natural is integer range 1.. integer'
  last;
type string is array (natural range <>) of
  character;
```

```
type recstr(leng: integer range 1..133 : =120)
  is
  record
    str: string(1..leng);
  end record;
    x: recstr(leng=>39);
    y: recstr(leng=>15);
    z: recstr; 제한요소 없음. 레코드결정인자
      leng는 1에서 133까지의 값을 가질 수 있
      으며, 배열형 레코드구성요소 str은 최대
      133개의 구성원소를 취할 수 있다. 초기치
      로서 leng는 120의 값을, str은 1과 120
      의 한계값을 갖는다.
  begin
    z:=y;
    z:=x;
    x:=z--z.leng=39인가를 검사해야 함.
```

위에서 레코드변수 *z*에는 제한요소가 없으므로 *z*의 레코드결정인자 leng는 레코드 대입문에 의해서 값이 변할 수 있다. 이렇게하면 배열형 레코드구성요소 str의 한계값(즉, 첨자제한요소)도 변경시킬 수 있다. 따라서 종속변수 제한요소의 범위內에서 배열변수의 크기가 변할 수 있어야 한다.

위에서 다소 예외적인 경우지만 일단 레코드변수를 생성한 후에도 레코드 제한요소에 의해서 크기가 변할 수 있는 동적배열변수를 예로들어 보았다. 반면 제한요소를 변경시키는 대입문을 성분변수(component variable)의 재생성으로 간주할 수 있다. 이 재생성 관점은 可變要素(variant)를 변경시키는 레코드 대입문의 경우 특히 편리하다. 왜냐하면, 그러한 변화는 舊可變要素에 포함되었던 성분요소를 없애버리고 新可變要素에 속하는 성분요소를 생성하는 것이기 때문이다.

제한요소가 있는 데이타 형식(위에서 *x*와 *y*)을 기억장소에 표현할 경우, 기억장소의 효율적 사용을 고려할 수 있다. 이때 대입문에서 제한요소가 있는 레코드와 제한요소가 없는 레코드를 혼합해서 사용할 수 있다는 점을 주의해야 한다.(물론 필요한 경우 범위의 일치성을 검사해야 한다.)

3. heap변수와 제한요소

access형식인 경우, 첨자와 결정인자 제한요소는 실행시에 new문장에 의해서 제공되어야 한다. 예를들면, 다음과 같다.

```
type accessstr is access string;
p: accessstr;
begin
  p:=new string (1..3000);
  p:=new string(m..n);
```

일반적으로 heap변수에 대한 종속형식 제한요소는 어떤 한 변수에 제한되어야 하는데, 이렇게 함으로써 특정변수를 가르키는 pointer에 의해서 이러한 정보를 인지할 수 있다.

4. 종속형식 제한요소와 부프로그램의 假媒介變數(formal parameter)

제한요소가 없는 배열의 假媒介變數인 경우, 假媒介變數의 한계치는 實媒介變數(actual parameter)의 값을 취하게 된다. 따라서 부프로그램을 수행하는 동안 實媒介變數의 한계치를 적용하면 된다. 제한요소가 없는 레코드의 假媒介變數인 경우 假媒介變數의 초기치는 實媒介變數에서의 레코드 결정인자의 값을 취한다. 만일 實媒介變數가 제한요소인 경우라면 부프로그램을 수행하는 동안 假媒介變數 레코드 결정인자의 값은 변할 수 없다.

5. 레코드 결정인자와 결정인자 제한요소

초기 Ada에서는 레코드 결정인자는 배열한계나 가변요소의 선택치로서만 사용할 수 있었지만, 개정된 Ada에서는 같은 레코드의 다른 구성요소에서도 결정인자 제한요소를 사용할 수 있다. 이러한 기능은 결정인자 제한요소로 레코드를 구성할 수 있음을 나타낸다. 왜냐하면 외부 레코드의 제한요소가 레코드 결정인자를 통해서 내부 레코드의 제한요소로 사용될 수 있기 때문이다. 이러한 기능때문에 동일한 외부 레코드형식을 갖는 두개의 변수가 내부 레코드 구성요소에서 서로 다른 내부 레코드 구성요소를 가질 수 있다. 예를들면 다음 예에서 변수 ats와 bts는 레코드 구성요소 sone에서 서로 다른 제한요소를 가지고 있다.

```
type multstr(n1, n2: integer range 1..512) is
  record
    sone: recstr(leng=>n1);
    stwo: recstr(leng=>n2);
  end record;
    ats: multstr(n1=>39, n2=>23);
    bts: multstr(n1=>15, n2=>23);
```

Ada의 레코드 결정인자와 결정인자 제한요소 기능은 매개변수형식으로 생각할 수 있는데 이러한 매개변수化는 동일한 데이타 형식의 서로 다른 변수로 하여금 서로 다른 可變要素와 서로 다른 크기의 배열형 구성요소를 가질 수 있도록 한다.[5]

Ⅲ. 데이타 형식과 변수의 표현

1. 데이타 형식과 종속형식의 표현

앞에서 설명한 대로, Ada의 레코드 결정인자와 결정인자 제한요소를 데이타 형식의 매개변수化로 생각할 수 있다. 원시 프로그램에 나타나는 숫치형식을 제외한 각 데이타 형식은 실행시 데이타 형식 기준표(type template)에서 설명되고 있는데 이러한 데이타 형식 기준표는 형식구별란에 의해서 배열, 레코드, 악세스형식이 구별된다. 배열이나 레코드 형식에 대한 기준표에는 假媒介變數란이 있는데, 레코드형식 기준표의 假媒介變數는 해당 레코드의 결정인자가 되며, 배열형식 기준표의 假媒介變數는 배열한계와 레코드를 구성하고 있는 구성원소의 데이타형식에 대한 매개변수이다.

원시프로그램에서 사용되는 데이타형식에 제한요소를 첨가할 수 있으며, 실행시에 이러한 제한요소가 데이타형식기준표의 實媒介變數가 된다.

제한요소 구별	(1) 제한요소 있음 (2) 제한요소 없음 (3) 외부매개변수에 의한 제한요소
제한요소	(1)인 경우, 제한요소의 수치값 (2)인 경우, 해당 데이타 형식설명표의 주소, 해당 데이타형식을 알 수 없는 경우는 공백. (3)인 경우, 외부매개변수의 참자번호. 예를들면, 외부변수의 세번째 매개변수.

그림 1. 제한요소표

데이타형식의 실매개변수는 제한요소표에 나타난다(그림 1). 제한요소표는 세가지 경우의 실매개변수를 취급한다 : (1) 제한요소가 있는 경우, (2) 제한요소가 없는 경우, (3) 외부변수의 형식매개변수에 의한 제한요소가 있는 경우. 제한요소표에는 두개의 란이 있는데, 제한요소 구별란은 위의 세가지 경우를 구별해 주고 제한요소란은 각각의 경우에 따라서 해당되는 값을 갖는다.

데이타형식의 실매개변수로 사용되는 모든 첨자나 레코드 제한요소는 인용표(call block)에 기억된다. 인용표는 실매개변수 기록표로 생각할 수 있다. 또한 인용표에는 형식기준표의 주소가 기록되어 있으며 실매개변수의 수가 기록되어 있는 란도 있다. 따라서 첨자나 레코드 결정인자 제한요소에 의한 종속형식은 인용표에 의해서 표현될 수 있다.

레코드는 여러개의 구성요소를 가질 수 있는데 각 성분요소는 고유한 데이타 형식을 가지고 있다. 레코드형식의 형식기준표는 각 구성요소에 대해서 구성요소의 형식기준표 혹은 인용표의 주소를 가지고 있다. 배열형식 기준표는 구성요소의 형식에 대한 형식기준표 혹은 인용표의 주소를 가지고 있다.

이러한 데이터형식의 매개변수化에 의해서 다음과 같은 장점을 얻을 수 있다.

(1) 매개변수化와 매개변수의 전달을 쉽게 처리할 수 있다. 전달형 데이타형식(composite type)의 경우, 실매개변수는 상위의 데이타 형식으로부터 하위인 구성요소의 데이타형식으로 전달된다.

(2) 제한요소가 없는 데이타형식을 취급. 다른 실매개변수가 전달되듯이 제한요소가 없다는 사실도 특별한 매개변수로 취급해서 전달할 수 있으므로 정확한 크기의 기억장소를 할당할 수 있다.

(3) 다른 구성요소에서 레코드 결정인자 제한요소로 사용되는 결정인자를 취급할 수 있다.

(4) B의 제한요소가 A의 매개변수가 아닐 때 A에 포함되어 있는 매개변수化된 B를 취급할 수 있다. 예를들면,

type A is record X: B (3,m); end record;

(여기서 변수 m의 값을 외부에서 전달된다.)

2. 변수의 표현

각 배열변수 혹은 레코드변수는 제한요소에 대한 정보를 포함하고 있다. 각 배열변수는 첨자제한요소(한계값)을 포함하고 있으며, 각 레코드 변수는 결정인자 정보를 포함하고 있고 또한 1개의 bit로 구성된 제한요소의 유무를 알려주는 란도 포함하고 있다.

배열과 레코드 변수의 표현은 고정부분과 가변부분의 두개의 부분으로 구성된다. 고정부분은 컴파일시 크기를 결정할 수 있으며 따라서 컴파일시 각

당한 크기로 기억장소를 할당할 수 있다. 가변부분의 크기는 실행시에 결정할 수 있는데 어떤 변수의 고정부분은 그 변수의 가변부분에 대한 주소를 포함하고 있다.

어떤 변수의 주소를 가르키는 값으로서는 절대주소를 사용하지 않고 상대주소를 이용해서 표시한다. 레코드의 구성요소들은 연속적으로 기억장치에 할당되는데 레코드의 구성요소들에 대한 고정부분 바로 다음에 가변부분이 위치하게 된다. 이러한 표현방법은 대입문의 경우 구역복사(block copy)를 사용하기 위함이다. 이러한 방법은 Ada Language Reference Manual[1]과 Rational[4]에서 제안한 방법이다.

3. 수치 종속형식

수치 종속형식에 대한 설명표는 그림 2와 같다. 수치형식인 경우에도 같은 모양의 설명표를 사용한다. 구별란은 정적수치와 동적수치(dynamic scalar)를 구별해 주며 동적수치인 경우 값의 범위는 실행시 결정된다(해당변수에 4bytes를 할당한다.). 정적수치인 경우, 값의 범위는 컴파일시 결정되며 할당되는 기억장소의 크기는 값의 범위를 표시할 수 있는 크기로 결정해 준다.

구별란 : 정적 수치 혹은 동적 수치
하한값 : 가장 작은 수치값
상한값 : 가장 큰 수치값

그림 2. 수치종속형식 설명표

4. 배 열

배열형식의 설명표는 그림 3과 같다.

표의 구분 : 배열기준표
매개변수의 수 : 매개변수의 수를 기록
차 원 : 차원의 수를 기록(예, 3차원)
구성요소 : 구성요소에 대한 형식설명표의 주소
첨자형식 : 각 차원에 대한 형식설명표의 주소 (차원의 수만큼 반복된다.)

그림 3. 배열형식 설명표

배열원소는 고정부분과 가변부분으로 구별된다. 고정부분은 그림 4에서와 같이 전형적인 배열 설명표이다. 가변부분에는 구성요소의 값들이 기억된다. 0첨자 주소란은 0번째 원소가 존재한다고 가정했을 때의 0번째 원소의 주소가 기록되는 란이다. n차원 배열인 경우, n-1개의 승산인자를 기억시키는데 여분의 승산인자란에는 각 구성요소의 크기를 기억시킨다.[5]

표의 구분 : 배열 변수
가변부분의 주소 : 가변부분의 주소를 기록
차 원 : 차원의 수(예, n 차원)
구성원소 : 구성원소의 총수
하 한 값 : 상 한 값 : } 차원의 수만큼 반복된다. 승산 인자 :
0첨자 주소 : 첨자가 0인 원소(가상원소임)의 주소

그림 4. 배열변수의 고정부분

5. 레 코 드

레코드 형식의 설명표는 그림 5와 같다.

표의 구분 : 레코드 기준표
매개변수의 수 : 매개변수의 수를 기록
고정요소의 수 : 고정형 구성요소의 수
가변요소의 수 : 가변형 구성요소의 수(0의 값을 갖으면 다음 두 란은 의미가 없음.)
가변요소 선택자 주소 : 가변요소의 구조를 결정해 주는 가변요소 선택자에 대한 정보를 포함하고 있는 기억장소
선택자 첨자번호 : case 문장에 적용할 매개변수의 첨자번호
구성요소의 데이타 형식 : 각 구성요소의 데이타 형식에 대한 설명표의 주소 (구성요소의 수만큼 반복)
가변요소의 구성요소 형식 : 각 가변요소를 구성하고 있는 요소에 대한 인용구역의 주소. 인용구역에서는 해당하는 요소의 레코드형식 설명표의 주소를 포함하고 있다.(가변요소의 구성요소 수만큼 반복)
매개변수 적용 : 각 매개변수에 대해서 해당매개변수가 적용될 구성요소의 위치

그림 5. 레코드형식 설명표

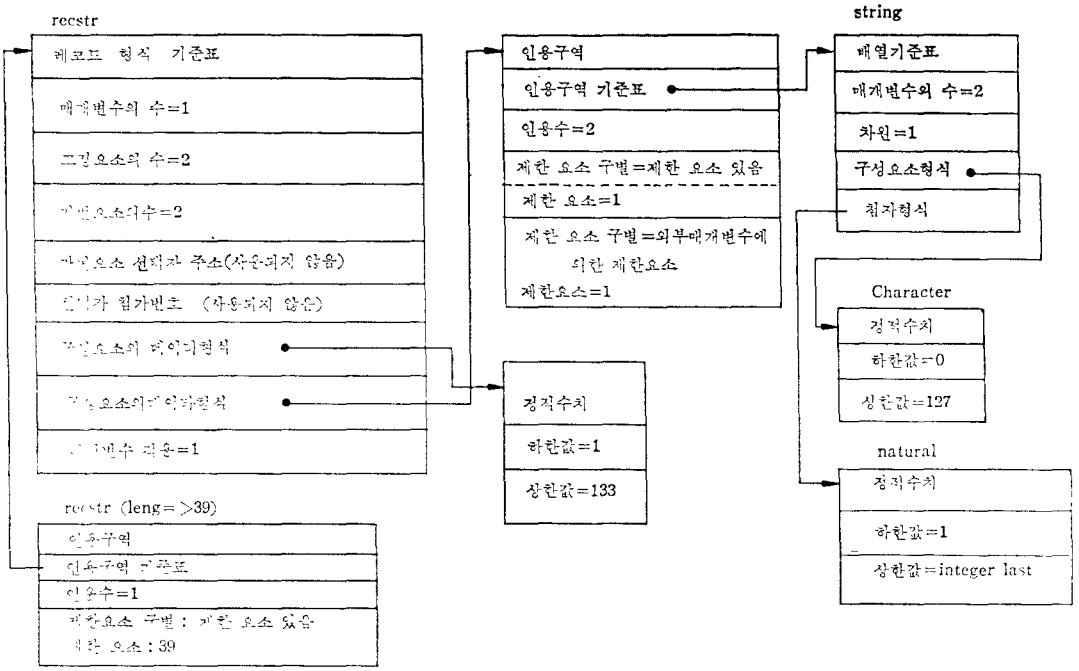

그림 6. recstr과 string에 대한 형식기준표 및 인용구역

앞의 예제에 있는 레코드형식 recstr에 대한 레코드형식 기준표를 나타내면 그림 6과 같다. 또한 recstr(leng=>39)의 결정인자 제한요소에 의해서 생성되는 인용구역과 string형식에 대한 배열형식 기준표도 나타내었다.

가변형 레코드는 각 가변요소에 대하여 독립된 레코드형식 설명표가 있는 것처럼 구성한다. 상위의 레코드 설명표에는 이러한 레코드 설명표의 주소를 포함하고 있는 인용구역의 주소를 포함시킨다. 상위 레코드의 결정인자(매개변수)가 가변요소의 내부에서 사용되도록 하면 인용구역은 여러 곳에 분산되어 나타나게 된다.

표의 구분 : 레코드 변수

제한요소 : 제한요소의 유·무

가변부분의 크기 : 레코드변수에서 가변부분의 총 크기(대입문에 의해서 변하지 않는다.)

가변부분 주소 : 가변부분의 주소

그림 7. 레코드 변수의 고정 부분

레코드변수는 고정부분과 가변부분으로 구분된다. 고정부분은 그림 7과 같으며 가변부분의 주소를 포함하고 있고, 가변부분은 그림 8과 같으며 가변부분에 대한 기억장소는 연속적으로 할당된다. 가변부분은 다시 고정부분과 가변부분으로 구별되는데 고정부분 다음에 가변부분이 온다. 단일 구성요소에 가변부분이 있다면 고정부분은 가변부분의 주소를 포함하게 된다. recstr형식의 레코드 변수를 표시하면 그림 9와 같다.

표의 구분 : 레코드 변수 가변부분

현재 크기 : 가변부분의 현재 크기, 대입문에 의해서 변경될 수 있음.

현재의 가변요소 : 현재 유효한 가변요소

각 구성요소의 고정부분 : (구성요소 수만큼 반복)

각 구성요소의 가변부분 : (구성요소 수만큼 반복)

그림 8. 레코드 변수의 가변부분

레코드의 가변요소를 구성하고 있는 요소들은 독립 레코드로 취급한다. 이 독립 레코드 內에 또다른 레코드 형식이 존재하게 되면, 이 레코드들도 독립 레코드로 취급하면 되므로 몇번이고 계속해서 이러한 레코드들이 존재하더라도 쉽게 문제를 해결할 수가 있다.

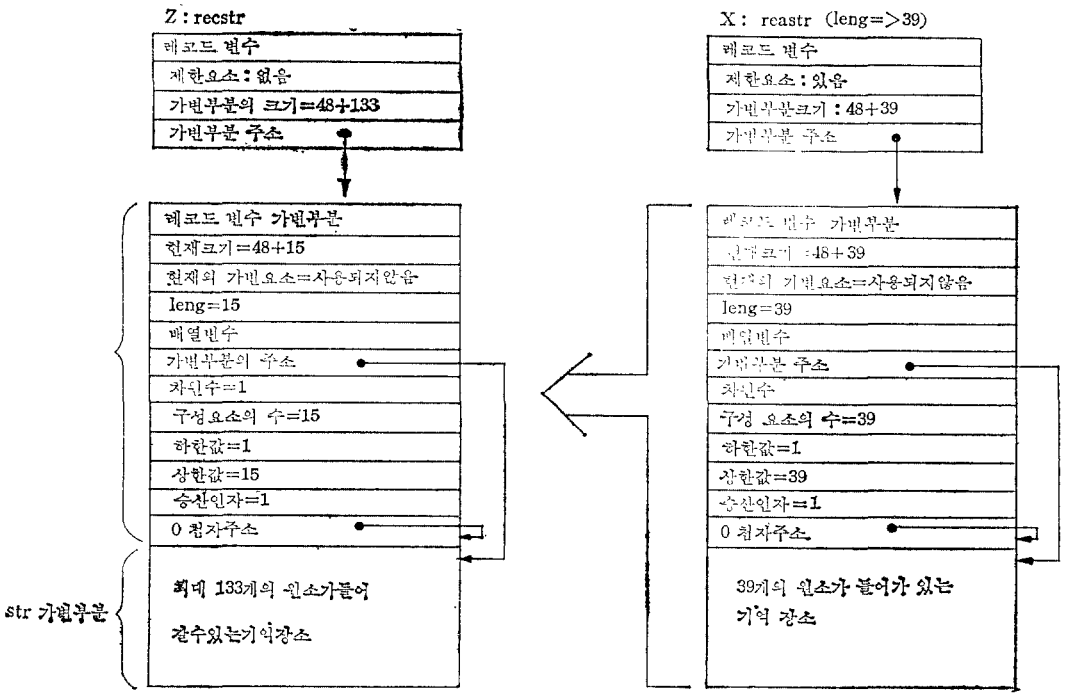

그림 9. 제한요소가 있는 레코드변수 x 를 제한요소가 없는 레코드변수 z에 대입하는 문장의 수행

6. 악 세 스

악세스(access)형 변수는 가변부분은 없고 heap에서의 주소를 포함하고 있는 고정부분만으로 구성된다. 악세스형 데이타형식 설명표는 악세스형임을 나타내는 표의 구분란과 악세스되는 데이타형식을 가르키고 있는 주소란으로 구성된다.

악세스 데이타형식에 대한 종속형식이 있는데 이는 악세스되는 heap변수에 대한 제한요소이다. 악세스 데이타형식에 대한 제한요소는 주어진 제한요소를 만족시키는 heap변수만을 악세스 할 수 있음을 의미한다. 컴파일러는 인용구역의 주소를 기억하므로써 이러한 종속형식을 표시할 수 있는데, 이 인용구역은 데이타형식 선언문에 의해서 구성된다. 이렇게 구성된 인용구역은 제한요소 검사를 요구하는 변수가 나타날 때마다 인용되게 된다. 컴파일러는 악세스되는 변수를 독립적으로 찾아가기 위해서 악세스형 변수와 인용구역을 둘다 이용한다. 예를들면, 실매개변수로서 악세스형 변수를 전달할 때, 전달하기 전·후에 제한요소 검사를 행해야 한다.

7. 실행시 기억장소의 할당

변수에 대한 기억장소의 할당시에는 데이타형식 기준표와 인용구역이 사용된다. 실행시의 기억장소 할당 프로그램은 변수의 동적부분(실제로 변수의 값이 기억되는 장소)을 할당하기 위해서 형식기준표와 인용구역을 참조하며 동적기억장소는 스택(stack)에 할당되게 된다. 할당프로그램은 변수의 고정장소에 설명표 정보를 기억시킴으로써 초기화시키며, 스택의 표현은 레코드변수의 표현과 유사하다. 예를들면, 어떤 프로그램에 배열변수가 있다면, 그 배열변수의 고정부분은 스택의 고정부분에 표시되며 스택內에서의 가변부분이 기억되어 있는 주소를 포함하고 있다.

Ⅳ. 결 론

본 논문에서는 Ada언어의 데이타 형식과 종속형식을 테이블을 사용하여 효과적으로 표현하였다. 이러한 방법을 이용하면 변수가 할당되는 기억장소에 관계없이, 즉 스택에 할당되어 있느냐, 혹은 heap에 기억되어 있느냐, 혹은 또 다른 변수의 요

소에 나타나느냐에 관계없이 동일한 방법으로 표현할 수 있다. 본 논문에서는 레코드 결정인자와 결정인자 제한요소를 매개변수화된 형식으로 취급하였는데, 이러한 매개변수화 개념에 의하면, "동일한 데이타형식을 갖는 두개의 변수가 서로 다른 가변요소와 서로 다른 크기의 배열요소를 가질 수 있다."라는 Ada데이타 개념을 쉽게 처리할 수 있다.

〔후 기〕

본 논문은 1982년도 문교부 학술조성 연구비의 도움으로 수행되었으며, 연구비 지급에 대하여 감사드린다.

참 고 문 헌

1. J.D. Ichbiah, J.C. Heliard, O. Roubine, J.G.P. Barnes, B. klieg-Brueckner, B.A. Wichmann. "Reference Manual for the Ada Programming Language." SIGPLAN Notices 14, 6 (June 1979), 1.
2. J.D. Ichbiah, B. krieg-Brueckner, B.A. Wichmann, H.F. Ledgard, J.C. Heliard, J.R. Abrial, J.G.P. Barnes, M. Woodeger, O. Roubine, P.N. Hilfinger, R. Firth. "Reference Manual for the Ada Programming Language. The revised reference manual", July 1980. edition, Honeywell, Inc., and Cii-Honeywell Bull, 1980.
3. Hilfinger, P.N. Discriminant Constraints in Ada, LIR. 008. Available on-line at ARPA-net site ISIE as 〈TNE-ARCHIVE〉 LIR. 008.
4. J.D. Ichbiah, J.C. Heliard, O. Roubine, J.G.P. Barnes, B. krieg-Brueckner, B.A. Wichmann. "Rationale for the Design of the Ada Programming Language." SIGPLAN Notices 14,6 (June 1979), 1.
5. D. Gries. Compiler Construction for Digital Computers. John Wiley and Sons, Inc., 1971.