!Null 널(NULL)의 활용

Null

Null이라는 것은 흔히 듣고 사용하는, "내용 없음, 비어 있음"을 의미합니다. IDL은 8.0에서부터 !null 이라는 시스템 상수로 Null 데이터형을 지원합니다. Null이 없이도 그 동안 별 문제 없이 지냈겠지만, 이왕 있는 건데 활용할 수 있다면 프로그램이 한두 줄이라도 더 간결해지고, 읽기 쉬운 코드가 될 것입니다.

Null의 형체

IDL 8.0 이전에는 다음과 같이 비어있는 배열이나 구조체가 용납되지 않았습니다.

```
IDL> a=[]
a=[]
   ^
% Syntax error.
```

문법 에러(Syntax error)를 내는 것을 보면 자료 구조뿐 아니라 문법적으로도 이런 표현은 허용이 되지 않았습니다. Null을 인정하는 IDL 8.0 부터는 다음과 같은 선언이 가능해 집니다.

```
IDL> b={}
b={}
   ^
% Syntax error.
```

```
IDL> a=[]
IDL> b={}
IDL> c=!null
IDL> help, a, b, c
A               UNDEFINED = !NULL
B               UNDEFINED = !NULL
C               UNDEFINED = !NULL
```

빈 배열(a=[ ]) 과 빈 구조체(b={ })를 선언했고 C의 경우는 !Null이라는 시스템 상수를 이용해 선언했는데, 이들의 데이터 타입은 모두 정의되지 않은 !null로 같습니다. 사실 위와 같이 선언을 해도 a를 앞으로 배열로 써야 한다든지, b를 구조체로 써야 한다는 제약은 없습니다. 아직 이들은 어떻게 쓰일지 운명이 결정되지 않은, 말 그대로 Null 상태입니다. 위의 세가지 선언은 완전히 같은 것입니다.

```
IDL> print, a eq b, b eq c, c eq a
   1   1   1
```

[ ]나 { }, !null이 모두 같은 의미임을 확인하였습니다.

(참고) 배열 이어 붙이기

IDL에서 두 개(또는 몇 개든)의 배열을 이어 붙이는 방법은 다음과 같습니다.

```
IDL> first=[0.5, 1.2, 3.3]
IDL> second=[100, 101, 102]
IDL> concat=[first, second]
     ;1차원 배열로 이어 붙일 때는 이렇게 합니다.
IDL> print, concat
    0.50   1.2   3.3   100.0   101.0   102.0
IDL> help, concat
CONCAT          FLOAT     = Array[6]
IDL> concat2=[[first], [second]]
;2차원 배열로 이어 붙일 때.
;[]를 중첩하여 배열의 차수를 지정합니다.
IDL> print, concat2
    0.500000      1.20000      3.30000
     100.000      101.000      102.000
IDL> help, concat2
CONCAT2         FLOAT     = Array[3, 2]
```

크기를 미리 알 수 없는 배열 늘려가기

크기가 얼마가 될지 알 수 없는 배열을 반복문으로 늘려 나가는 구문을 만들 때, Null이 없던 버전에서는 몇 가지 방법이 있었습니다. 조금은 귀찮은 방법이지만,

1. 처음에 배열에 아무값을 하나 넣고, 거기에 이어서 요소들을 붙여 나간다. 마지막에 배열의 0번(아무값)을 제외하고 추출한다. 즉, 배열 1번부터 끝번까지만 뽑아 낸다.
2. 반복문 안에 if 문을 사용하여 배열의 개수가 0일 때는 새로 배열을 생성하도록, 아닐 경우에는 기존 배열에 붙여 나가도록 구문을 작성한다.
3. 충분히 큰 배열을 미리 생성해 놓고 값을 대체해 나간다. 마지막에는 처음부터 실제 사용한 끝 요소까지 추출한다.

!NULL을 이용한 배열 초기화

Null을 사용하면 이 과정이 조금 단순해집니다. 처음에 배열을 Null로 초기화하고 이 배열을 계속 이어나가는 방법을 쓰면 됩니다.

```
IDL> myarr=[]  ;또는 myarr=!null
IDL> for i=0, 9 do myarr=[myarr, i]
IDL> print, myarr
  0   1   2   3   4   5   6   7   8   9
```

!NULL을 이용한 구조체 초기화

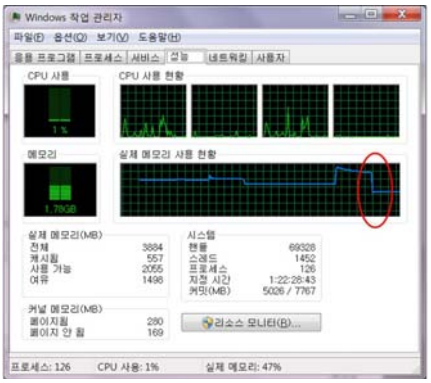
구조체를 생성하는 CREATE_STRUCT( ) 함수도 배열 이어 붙이기와 비슷하게 기존 구조체에 필드를 계속 추가해 나가는 문법을 지원합니다. 필드 수를 미리 알수 없는 상황에서 반복 구문을 이용하여 구조체를 확장해 나갈 때, 초기값으로 !NULL을 사용한다면 구문을 조금 더 간결하게 만들 수 있습니다.

!Null을 이용한 변수의 해제

사용하던 변수에 !null 상수를 대입하면 해당 변수의 메모리를 해제하는 효과가 있습니다.

```
IDL> testvar=fltarr(20000, 20000)
  ;약 1.6기가 바이트의 배열.
IDL> testvar=!null
```

작업관리자를 통해 배열 선언이후 메모리 사용량과 메모리 해제 후의 사용량 변화를 보면 !null이 변수의 메모리 해제에 이용될 수 있음을 확인할 수 있습니다.



!null로 변수를 해제한 이후 메모리 변화

!Null과 비교를 통한 변수 상태 확인

!null을 변수 또는 포인터와 비교하면 변수에 값이 있는지 여부를 판단하거나 포인터에 값이 있는지 여부를 판단하는 데에 유용합니다.

```
IDL> print, var_not_defined eq !null
;결과는 1. 즉, var_not_defined는 아직 사용되지 않는 변수
IDL> print, ptr_new() eq !null
;결과는 1. 널 포인터의 경우 !null과 같다고 판단함
IDL> defined=['abc']
IDL> print, defined eq !null
;결과는 0. 값이 있는 변수는 !null과 같지 않음
```

Where() 함수와 !Null

Where() 함수는 조건에 맞는 요소가 없는 경우 -1을 리턴합니다. 8.0 버전 이전에는 배열의 인덱스로 음수(negative)를 사용할 때 에러를 냈습니다. 8.0부터는 음수로 배열을 조회할 경우 배열의 맨 뒤에서부터 조회하는 편리한 기능이 추가되었습니다. 단, where() 함수의 리턴값 -1(해당 요소 없음)을 잘못 해석하여 배열의 맨 끝 요소를 찾게하는 오류를 범할 가능성이 생겼습니다. 가장 안전한 방법은 where() 함수의 count 파라메터를 이용하여, 조건에 일치하는 배열 개수가 몇 개인지 세고, 0개 일 경우에는 별도의 조치를 취하는 것입니다.

```
IDL> a=indgen(10)
IDL> ok=where(a gt 100, count)    ;해당 요소는 없음
IDL> print, ok
  -1
IDL> print, a[ok]   ; 즉, a[-1]
; IDL 8.0이전에는 에러. IDL 8.0 부터는 a 배열의 마지막
; 요소인 9. 어느쪽이든 원하는 결과는 아님
IDL> print, count
   0
;즉, 조건에 부합하는 배열 요소는 없음
IDL> if count gt 0 then print, a[ok]
;항상 WHERE() 함수 조건에 일치하는 요소가 0개인지
;확인하는 것이 안전
```

!Null의 등장과 함께 Where() 함수에서도 /NULL 키워드가 추가되었습니다. 이 키워드를 사용하면 조건에 일치하는 배열 요소가 없을 경우 리턴값을 기존의 -1 대신 !null로 대체합니다.

```
IDL> ok=where(a gt 100, /NULL)
IDL> print, ok
 !NULL
IDL> print, a[ok]
 !NULL
```

a[!null]의 결과는 !NULL이므로 에러를 내지도, 배열의 맨 마지막 요소를 리턴하는 혼란을 야기하지도 않습니다. /NULL 키워드를 사용하면, COUNT가 0인지 비교해 보지 않아도 안전한 코드를 생성할 수 있습니다.

사용하지 않을 리턴 변수를 대체하는 !NULL

!NULL 상수를 대입문의 좌변에 사용할 수도 있지만 그 결과는 아무런 것이 없습니다.

```
IDL> !null=3*4
```

위와 같은 문장이 오류 없이 처리는 되지만 결과적으로는 아무런 의미가 없습니다. 이 특성을 활용하여, 함수의 리턴값을 사용하지 않고자 할 경우 좌변을 !NULL로 적용할 수 있습니다. 잘 사용되는 기능은 아닙니다만 사용하지 않을 값이라는 의미는 명확히 표현할 수 있습니다. LABEL_DATE() 라는 함수의 리턴값은 원래 의미가 없습니다(항상 0을 리턴합니다).

```
IDL> dummy=label_date(date_format='%Y/%M/%D')
```

이렇게 사용해도 좌변의 dummy 변수에는 언제나 0이 입력되며 사실상 사용될 일이 없습니다. 그러므로 다음과 같이 !null을 이용하여 리턴값이 사용되지 않을 것임을 코드상에서 명확히 표현할 수도 있습니다.

```
IDL> !null=label_date(date_format='%Y/%M/%D')
```