FORTRAN ve Komutlarının Kısa bir Tekrarı

Bu kısa hatırlatma notunda herhangi bir FORTRAN programlama dili dersinde görülen komut ve özelliklerin tekrarı yapılmıştır. Daha fazla detay gerektiğinde bir FORTRAN kitabına başvurulması gerekir.

1 FORTRAN: Kısa Bir Tarihçe

FORTRAN en eski programa dilleriden biridir. Adı 'FORmula TRANslation' veya 'FORmula TRANslator' (Formül Tercümanı) kelimesinden türetilmiştir. İlk Fortran (hem FORTRAN hem de Fortran yazım şekilleri yaygın olarak kullanılmaktadır) derleyicisi 1954-57 yılları arasında IBM'de çalışan bir grup programcı tarafından yazılmıştır. Bu dönemlerde bilgisayarlar çok az bir hafıza kapasitesi (15 KB civarı) sahip olup ve işletim sistemleri ilkel ve çok yavaş idi. Bu ilk derleyici FORTRAN I olarak adlandırıldı ve çok kısa sürede beğenilerek yaygınlık kazandı, bilimsel çevrelerde kullanımı arttı. Fortran'ın ilk kurucuları görünüşte insan diline yakın bir bir programlama dili oluşturmak istediler.

Daha sonra FORTRAN II (1958), III (1958), IV (1961) ve 66 serileri izledi. FORTRAN III hiç bir zaman piyasaya sürülmedi. 1962 yılında FORTRAN için bir standart yazılmaya başlandı ve 1966 yılında bu standart yayınlandı, bu nedenle FORTRAN 66 olarak bilinir oldu. Bu dünyadaki ilk Yüksek Seviyeli Dil standardıdır.

Her yeni sürüm eskisine nazaran daha iyi özelliklerle kullanıcılara sunuldu. FORTRAN'ın çok başarılı olmasının bir sebebide artık programlamanın basit olmasında yatar. Daha sonra FORTRAN 77 piyasaya sürüldü, çok eski olmasına rağmen günüzümüzde hala bu standartta yazılmış programlar derlenmekte ve kullanılmaktadır. Fortran derleyicilerinin başka önemli bir özelliği ise geriye doğru uyumlu olmalarıdır. FORTRAN 77 ile programlama diline bir çok yeni özellikler eklendi. Yeni veri tipleri, döngüler, karar verme süreçleri, kontrol devri komutları bu yenilikler arasındadır.

FORTRAN 77'den uzun yıllar sonra Fortran 90 standardı yayınlandı. Bu kadar uzun bir ara verilmesi C gibi başka dillerin ortaya çıkmasına ve gelişmesine yol açtı. Daha sonraları Fortran 95, Fortran 2008 standartları da eklenerek Fortran dili gelişmesine devam etti.

Fortran günümüzde bilimde, mühendislikte, uzay çalışmalarında, paralel programlama gerektiren büyük ölçekli çok yoğun hesaplamalarda, askeri uygulamalarda kullanılan ve tercih edilen bir programlama dilidir.

2 Karakter Sayısı

FORTRAN programa dilinde 49 tane karakter kullanılır. Bunlardan 26 tanesi latin alfabesinin A'dan Z'ye kadar olan harfleri ve 10 tanesi 0'dan 9'a kadar olan rakamlardır. Geriye kalan 12 sembol çoğunlukla [ ] ile temsil edilen boşluk ve şu karakterlerden oluşur: + - * / = ( ) $. , ' : Alfabe harfleri ve rakamlardan oluşan karakterler alfasayısal karakterler olarak adlandırılır.

3 Değişkenler

Değişken, bilgisayarda belirli bir hafıza birimine verilen sembolik bir isimdir. Bir alfabe harfi ile başlaması gerekir. Diğerleri alfasayısal karakterlerden oluşabilir ve bir değişken ismi en çok 31 karakterden meydana gelir. Değişkenler tamsayı veya kesirli olabilir. Başka türlü belirtilmediği takdirde I, J, K, L, M ve N harfleri ile başlayan değişkenler tam sayı olarak kabul edilir. örneğin N2, MAX, I89, KAZ tam sayı değişkenlerdir. A, B, ..., H, O, P, ..., Z ile başlayan değişkenler ise başka türlü belirtilmediği taktirde gerçel (reel) kabul edilirler. X1, SAZ, F1, P2W reel değişkenlere örnektir.

4 Sabitler

Bir FORTRAN programının çalışması sırasında değişmeyen tam sayı veya reel sabitlerdir. Bunların dışında bir sabit sanal (kompleks), karakter veya mantıksal olabilir. 9, 11, 32 476 sabit tam sayılara, 3.15, 4.2, 0.0032 reel sayılara örnektir. Reel sayılar exponansiyel temsil ile de gösterilebilir. örnegın 0.0032 yerine 3.2E-3 yazılabilir. Sanal sabitler ise (a,b) şeklinde temsil edilirler. Burada a sayının gerçel, b ise sanal kısmını gösterir. örneğin $2.0 + 7.0i$ kompleks sayısı (2.0,7.0) şeklinde gösterilir. Mantıksal sabitler iki tanedir: .TRUE. ve .FALSE. Son olarak karakter sabitler 'Karakuyu 1341' örneğinde olduğu gibi apostroflar arasında yazılarak gösterilirler

5 Tanımlama Komutları

Bir programda kullanılan değişken veya sabitlerin tiplerinin (reel, tamsayı vb.) açık olarak belirtilmesinde fayda vadır. Eğer sabit veya değişkenler onların kabul edilen tiplerinden farklı tipleri temsil ediyorlarsa mutlaka belirtilmesi gerekir. örneğin IZ4 değişkeni bir programda reel bir sayıyı temsil ediyorsa bunun mutlaka belirtilmesi gerekir. Tanımlama komutları aşağıda birer örnek ile gösterilmiştir.

```
INTEGER A, IC, KZ, BIZ
REAL KL, RA, YA
REAL*8 PR, IP, SX
COMPLEX J, ID, X2
COMPLEX*16 Z, IH, DE, E
DOUBLE PRECISION AZ, IDA, X2Y
LOGICAL K, GA, DX
```

Ayrıca boyutları olan bir değişken varsa onun tanımlanması gerekir. Aşağıda verilen örnekleri göz önüne alalım.

```
DIMENSION A(5), B(0:7), C(-4:5), D(3,4), E(0:5,0:6)
```

A, B ve C değişkenleri bir boyutlu, D ve E ise iki boyutlu değişkenlerdir. A'yı oluşturan elemanlar A(1), A(2),... A(5)'dir. B'nin 8 elemanı vardır: B(0), B(1), ..., B(7). C'nin C(-4),C(-3),..., C(0), ..., C(5) olmak üzere 10 tane elemanı vardır. D'nin toplam 3×4=12 elemanı, E'nin ise 6×7=42 elemanı vardır.

Bazı değişken veya sabitlerin değerleri programın başlangıcında atanmak isteniyorsa DATA komutu kullanılabilir.

```
DATA A,B,C,IC/10.,-2.E-3,1.0,5/
DATA A/10./B/-2.E-3/C/1.0/IC/5/
DATA ADSOYAD/'METIN OZDEMIR'/
DATA (X(I),I=-1,5) /-1.,2.,5*4./
```

Yukarıdaki birinci ve ikinci satır tamamen eşdeğerdir. üçüncü satır karakter tipi olan bir değişkene aldığı değeri atamaktadır. Son satır ise birden fazla elemanı olan X vektörünün değerlerini atamaktadır. Bu satırda geçen '5*4', '4'ün beş defa kullanılacağını göstermektedir. Yani X(1)=X(2)=...=X(5)=4 anlamına gelir, burada $X(-1) = -1$ ve $X(0) = 2$'dir

Aynı parametre veya değişken bir veya birden fazla alt program tarafından kullanılıyorsa, bu değerler COMMON komutu da kullanılarak gereken yerlere transfer edilebilir. Bunun adlı ve adsız (veya boş) olan iki çeşidi vardır. Eğer adsız COMMON kullanılırsa, bir programda sadece bir COMMON komutu kullanılabilir. Adlı COMMON kullanılırsa programcı istediği kadar COMMON komutu kullanabilir. Aşağıda birinci satır adsız, diğerleri ise adlı COMMON komutlarının kullanımına örneklerdir.

```
COMMON A,B(5),C(-1:3,0:8)
COMMON/PAR1/ D,E(3),F(10)
COMMON/PAR2/ HBAR,QE,PI,KBT
COMMON/PAR3/ X(NMAX),Y(NP)
```

6 Aritmetiksel Komutlar ve Matematiksel Fonksiyonlar

Aritmetiksel bir komut program derleyicisine aşağıdaki beş işlemciden herhangi birini gerektiren bir işlem yapmasını sağlar. Beş aritmetik işlemci:

+	toplama
-	çıkarma
*	çarpma
/	bölme
**	üs alma

komutlarından oluşur. Eğer parantezler kullanılarak özellikle tanımlanmamışsa aşağıda verilen öncelik sırası geçerlidir.

1. üs alma
2. çarpma ve bölme
3. toplama ve çıkarma

Örneğin komut satırına yazılan A**B/C*D-E ifadesi derleyici tarafından $\frac{A^B}{C}D - E$ şeklinde yorumlanır. Aşağıda matematiksel işlemlere örnekler verilmiştir.

```
I=I+1
X=A*X+X
B(I)= A(J)-B(K)**2
A= 2.*SIN(X) + 1.5
```

Burada SIN(X) FORTRAN derleyicisinde zaten tanımlanmış bir fonksiyondur. Kullanıcının ayrıca bir tanım yapmasına gerek yoktur. Aşağıdaki listede FORTRAN'da varolan tanımlı fonksiyonlar ve özellikleri verilmiştir.

[eklenecek]

7 Kontrol Devri

Programın kontrolü bir satırdan bazı şartların sağlanması durumunda başka bir satıra aktarılabilir. Bu durum programın akış mantığında bir değişiklik yaratır. Kontrol devri iki şekilde yapılabilir: şartsız veya şartlı devir. Şartsız devirde komut

```
GO TO n
```

şeklindedir. Burada n kontrolun devredileceği satır numarasına tekabül eder. Aşağıdaki örnek bu kullanımı göstermektedir. Dikkat edilirse burada programın kontrolu 10 numaralı satıra hiç bir şart aranmadan devredilmektedir.

```
      GO TO 10

        ⋮

10      X=2.*X + B
```

şartlı kontrol devri ise iki şekilde yapılabilir. Bunların birincisinde komut satırı

$$\text{GO TO } (n_1, n_2, \ldots, n_k), \text{ I}$$

şeklindedir. n_i'ler program işleyişinin devredileceği satır numarasını gösterir ve I sadece I=1,2,...,k değerlerini alabilir. örneğin

$$\text{GO TO } (10, 20, 90, 10), \text{ IC}$$

komutu IC=1 veya 4 olduğunda 10 nolu, IC=2 olduğunda 20 nolu ve IC=3 olduğunda 90 nolu satıra gidileceğini ifade eder. Yani IC=1 veya 4 olması durumunda 'GO TO 10' komutu varmış gibi işlem yapılır. Diğer durumlar içinde benzer kurallar geçerlidir. Diğer bir şartlı kontrol devri ise IF komutu ile yapılır. IF komutunun argümanı aritmetik veya mantıksal olabilir. Aritmetik argümanlı IF komutu

$$\text{IF}(\ terim\ )\ n_1, n_2, n_3$$

şeklindedir ve eğer $terim < 0$ ise n_1, $terim = 0$ ise n_2 ve $terim > 0$ ise n_3 nolu satıra gidileceğini ifade eder. Mantıksal IF komutu

IF(mantıksal terim) komut X

şeklinde olup mantıksal terimin doğru olması durumunda X komutunun yerine getirileceğini gösterir. örneğin

```
IF(B.LE.E) GO TO 32  [veya if(b =< e) go to 32 ]
```

komut satırı B'nin E'den küçük veya eşit olması durumunda 32 nolu satıra gidileceğini gösterir. Benzer şekilde

```
IF(A.LT.-1.AND.B.GT.3) X=X*X-1.0  [veya if(a < -1.and.b>3) x=x*x-1 ]
```

komutu A'nın -1'den küçük olması ve B'nin 3'den büyük olması durumunda X'in değerinin X'in karesinden 1 çıkartılarak elde edilen sayıya eşitleneceğini gösterir.

IF komutlarında kullanılan karşılaştırma komutları aşağıda verilmiştir.

.LT.	(<)	küçük
.LE.	(=<)	küçük veya eşit
.GT.	(>)	büyük
.GE.	(>=)	büyük veya eşit
.EQ.	(==)	eşit
.NE.	(><)	eşit değil

Mantıksal işlemciler ise

.AND.	ve
.OR.	veya
.NOT.	değil

ile verilir.

Bir diğer IF komutu ise 'IF(...) THEN ... ELSE ... ENDIF' komutudur. Genel kullanım şeklini bir örnek ile gösterelim:

```
        IF(N.EQ.1) THEN
        X=X + Y
        Z= SIN(X)
        ELSE IF(N.EQ.2) THEN
        X= X * Y
        H= F(X)
        ELSE
        X= X * X
        CALL TOPLA(X,Y,Z)
        ENDIF
```

ENDIF komutu blok halinde yazılan IF komutunun bittiğini belirtir. Burada N 1'e eşit ise ondan sonra gelen iki satır işlenir ve IF döngüsünden çıkılır. N 2'ye eşit ise o satırdan sonra gelen iki satır gözönüne alınır. Eğer N 1 veya 2'ye eşit değilse 'ELSE'den sonra gelen satırlar işlem görür ve döngüden çıkılır.

8 DO Döngüleri

DO döngüleri FORTRAN programlarında tekrar eden işlemleri yapmak için kullanılır. Genel kullanım şekli aşağıda verildiği gibidir.

```
        DO n I=NILK, NSON, NARTI

        (komutlar)
n       CONTINUE
```

Burada n DO döngüsünün numarasını belirtir. I herhangi bir tamsayı değişkenidir. NILK I'nın alacağı ilk değeri, NSON, I'nın alacağı son değeri ve NARTI ise I'nın değerinin kaçar kaçar arttırılacağını tanımlar. Aşağıdaki örnekte I'nın 1'den 20'ye kadar 3'er 3'er arttırılacağını gösterir.

```
        X=0.0
        DO 10 I=1,20,3
        X= X + SIN(X)
10      CONTINUE
```

9 Girdi-Çıktı Komutları

Bir programın çalışması için gereken veriler ekrandan doğrudan girilebileceği gibi verilerin daha önce yazıldığı bir dosyadan da okutmak mümkündür. Benzer şekilde programın ürettiği veriler ekrana yazılabileceği gibi herhangi bir dosyaya yazılarak uzun süreli olarak korunabilir. Bir dosyadan veri okumak veya dosyaya veri yazmak için o dosyayı açmamız gerekir. Bu OPEN komutu ile yapılır. Genel kullanım biçimi

```
      OPEN(n, FILE='DOSYAADI', STATUS='UNKNOWN')
```

şeklindedir. Burada n bir tam sayı olup birim numarası olarak bilinir. DOSYAADI, açılması istenen dosyanın adıdır ve daha önceden 'CHARACTER' tipi değişken olduğu belirtilmelidir. 'STATUS' dosyanın statüsünü belirtir. Statü 'OLD', 'NEW' veya 'UNKNOWN' olabilir. Statünün 'OLD' olması daha önceden varolan bir dosyanın açıldığı anlamına gelir. 'NEW' yeni bir dosyanın açıldığı, UNKNOWN ise açılmakta olan dosyanın statüsünün bilinmediği anlamına gelir. 'OPEN' ile açılan bir dosyadan veri okunarak veya dosyaya veri yazılarak dosyanın belirli bir satırına kadar gelinebilir. Dosyanın herhangi bir sebeple başına dönülmesi istenirse 'REWIND(n)' komutu kullanılır. Burada n geriye sarılması istenen ve daha önce 'OPEN' komutu ile açılan dosyanın birim numarasıdır. 'OPEN' komutu ile açılan bir dosya gerektiğinde, n açılan dosyanın birim numarası olmak üzere 'CLOSE(n)' komutu ile kapatılır.

ıki önemli girdi ve çıktı komutu READ ve WRITE komutlarıdır. Birincisi verileri ekrandan veya bir veri dosyasından okumaya, ikincisi ise elde edilen verileri ekrana veya bir dosyaya yazmaya yarar. Okuma ve yazma işlemleri formatlı olabileceği gibi formatsızda olabilir. Formatlı okuma veya yazmada çıktıların nasıl okunacağı ve yazılacağı net olarak tanımlanır. Formatsız okuma ve yazma komutları

```
      READ(*,*) liste
      WRITE(*,*) liste
```

şeklindedir. Liste burada okunacak/yazılacak değişkenleri temsil eder. Parantez içindeki birinci yıldız okumanın/yazmanın ekrandanekrana olacağını belirtir. Eğer yıldız yerine bir tam sayı kullanılırsa daha önce OPEN komutu ile açılan aynı sayı numaralı dosyadan/dosyaya okumayazma yapılacağı anlamına gelir. ıkinci yıldız yerine bir tam sayı kullanılırsa okumayazmanın bu numaralı formata göre yapılacağı anlamına gelir. Aşağıda incelemeniz için bir kaç örnek verilmiştir.

```
      READ(*,*) A, N
      WRITE(*,*)' X=',X,' Y=',Y
      READ(12,*) C,D,M
      WRITE(15,100) K,(A(I),I=1,3)
100   FORMAT(1X,'K=', I5, 1X,'A=',3(2X,E12.5))
```

10 Alt Programlar

Bir programda yapılacak işler parçalara ayrılarak her bir parçayı yapacak alt programlar yazılabilir. Her alt program kendi içinde bir bütünlük oluşturur. Alt programların en temel olan iki tanesi FUNCTION ve SUBROUTINE alt programlarıdır.

FUNCTION alt programının bir adı ve girdi olarak kullanılacak argümanları olur. Bu alt programın sadece bir çıktısı vardır ve programın kendi adı ile aynıdır. Aşağıda verilen örnek FUNCTION alt programı $f(x, y) = x^2 + 2xy - x * \sin((x + y)/2x)$ fonksiyonunun değerini verilen x ve y değerleri için hesaplar ve değerini F'ye kaydeder.

```
FUNCTION F(X,Y)
Z= (X+Y)/X/2.
F= X*(X + 2.*Y-SIN(Z))
RETURN
END
```

FUNCTION alt programı program içinde herhangi bir yerde adı ile doğrudan çağrılabilir. Örneğin yukarıda tanımlanan $f(x, y)$ fonksiyonunu değeri herhangi bir anda $x = 1.2$ ve $y = -3.5$ değerleri için hesaplanmak isteniyor ve sonuç 'a' değişkenine atanmak isteniyorsa aşağıda verilen örnekte gösterildiği gibi bu hesaplama yapılabilir. Benzer şekilde F(X,Y) fonksiyonu doğrudan hesaplama işlerinde de kullanılabilir. Aşağıda 'b' değikenine atanan değer gibi.

```
x=1.2
y=-3.5
a=F(x,y)
b= x*F(x,y) + y*F(2.*x,3.*y)
```

SUBROUTINE alt programı ise girdi olarak bir çok değer alabilir ve FUNCTION alt programından farklı olarak birden çok çıktı verebilir.