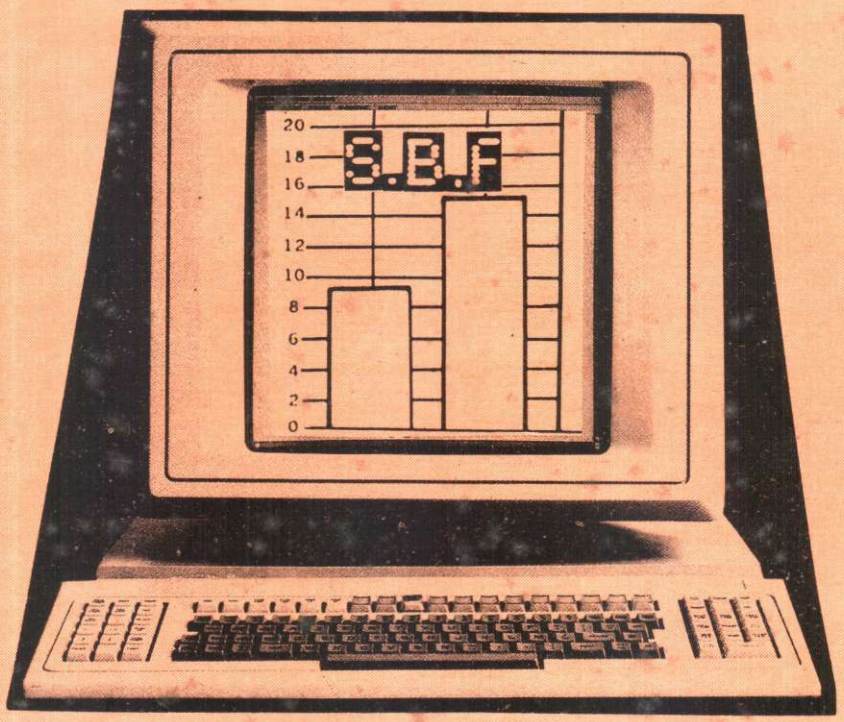
ANKARA ÜNİVERSİTESİ SİYASAL BİLGİLER FAKÜLETESİ YAYINLARI : 567

İŞLETMECİLİKTE FORTRAN IV İLE PROGRAMLAMA



Doç. Dr. Halil SARIASLAN
A.Ü. Siyasal Bilgiler Fakültesi
İşletme Bölümü

Gözden Geçirilmiş İkinci Baskı

Ankara 1988

ANKARA ÜNİVERSİTESİ SİYASAL BİLGİLER FAKÜLETESİ YAYINLARI: 567

İŞLETMECİLİKTE FORTRAN IV İLE PROGRAMLAMA

Doç. Dr. Halil SARIASLAN
A. Ü. Siyasal Bilgiler Fakültesi
İşletme Bölümü

Gözden Geçirilmiş İkinci Baskı

Ankara 1988

Babam Bilâl'a
ve
Annem Rabia'ya

ISBN 975–482–000–7

A.Ü. S.B.F. ve BASIN - YAYIN YÜKSEKOKULU BASIMEVİ, ANKARA - 1983

ÖNSÖZ

Çağımızın en belirgin teknolojik simgesi olan bilgisayarlar, günümüzde çok geniş bir kullanım alanı bulmuşlardır. Bunun bir sonucu olarak bilgisayar programlaması bugün birçok üniversitelerimizde eğitim programlarının vazgeçilmez bir parçası olmuştur. Ankara Üniversitesi Siyasal Bilgiler Fakültesinde okutulmakta olan Bilgisayar programlaması dersi için ders kitabı gereksinimini karşılamak amacı ile hazırlanan bu çalışma, işletmecilik ve iktisat alanındaki örneklerle FORTRAN IV programlama dilinin temellerini vermeye çalışmaktadır.

Çalışmanın bir programlama ders kitabı olarak çerçevesini taşırmamak için verilen programlama örneklerinde ele alınan yöntem ve tekniklerin yalnızca programlama ile ilgili yönleri belirtilmiştir. Ancak örnek programların anlaşılmasını kolaylaştırmak için programların dayandığı yöntemler, gerektiğinde sayısal örneklerle açıklanmıştır. Böylece, çalışmanın bir ders kitabı olarak amacını gerçekleştirmesi için gerekli olan açıklığa sahip olabileceği düşünülmüştür.

Bu kitabın hazırlanmasında, başlangıcından bitimine kadar pek çok kişinin değişik derecelerde katkısı olmuştur. İsim belirtmeksizin hepsine burada teşekkür etmeyi bir borç bilirim. Ancak SBF-BYYO matbaasında kitabın titizlikle basımını sağlayan görevlilere özellikle teşekkür etmeden geçemeyeceğim.

Kitabın okuyucularına yararlı olması tek dileğimdir.

Halil SARIASLAN — Ankara, 1983

İÇİNDEKİLER

ÖRNEK PROGRAMLAR

I.

BİLGİSAYARIN TEMELLERİ

Çağımızın en önemli teknolojik simgesi olan sayısal bilgisayarlar (digital computers) tarihi gelişimlerinin aksine çok hızlı bir biçimde geniş bir kullanım alanı bulmuşlardır. Endüstri ve iş dünyasındaki kullanımları yanında, tüm bilimsel çalışmaların ve üniversite eğitim programlarının da vazgeçilmez bir parçası olmuşlardır. İş ve bilim dünyasındaki başarılı işlevlerine parelel olarak, bilgisayarlar insanların günlük toplumsal yaşantılarına da girmişlerdir. Denilebilir ki, II. Dünya Savaşından bu yana bilgisayarlar kadar geniş bir kullanım alanı bulan teknolojik gelişmeler parmakla sayılacak kadar azdır. Uzayın keşfi çalışmalarından askeri harekatları değerlendirmeye, spor-toto tahminlerinden kadın ve erkekleri ideal eşleştirme! gibi çöp çatanlığa varıncaya kadar pek çok çeşitli işler için bilgisayarların kullanıldığını belirtmek bu araçların çağımızda ne denli önemli bir yer tuttuğunu açıklamaya yetecektir.

O halde, sayısal bilgisayar dediğimiz bu araçlar nedir? Bu soruyu yanıtlamadan önce hemen belirtelim ki, bilgisayarların diğer bir türü olan ve voltaj, akım, uzunluk, ısı, basınç, hız v.b. büyüklükleri ölçmek için düzenlenen "analog bilgisayarlar" konumuz dışındadır. Bundan sonraki sayfalarda konumuz olan sayısal bilgisayar sözcüğü yerine yinelemelerden sakınmak için yalnızca bilgisayar sözcüğü kullanılacaktır. Diğer bir deyişle, bilgisayar sözcüğünden sayısal bilgisayar anlaşılmalıdır.

1.1 Sayısal Bilgisayarların Tanımı ve Özellikleri

Kimi zaman elektronik beyin ya da bilgi işlem makinası olarak da adlandırılan bilgisayar, kendisine önceden verilen yönergelere (talimatlara) göre verileri elektronik olarak büyük bir hızla işleyen ve bir bellek sistemi olan otomatik bir makina diye tanımlanabilir (Davis, 1973).

Bu tanıma dayalı olarak bilgisayarın özellikleri aşağıdaki gibi sıralanabilir.

1. Veri işler ve bu işlemeyi kendisine önceden belli bir sistem aracılığı ile verilen yönergelere göre yapar.

2. Elektronik olarak çalışır.

3. Veri işlemeyi büyük bir hız ve doğrulukla yapar. Bir saniyede 250 binden fazla toplama, dakikada 25 milyon civarında çarpma işlemi yapabildiğini belirtmek bilgisayarın ne kadar hızlı işlem yaptığına bir örnektir.

4. Bir bellek sistemi vardır. Bilgisayar toplu iğne başı büyüklüğünde manyetik çekirdeklerden (magnetic cores) oluşan bir içsel saklama sistemine sahiptir. Bilgisayar bu içsel saklama sistemi aracılığı ile çok sayıda yönerge ve veriyi içinde tutabilir.

5. Veri işlemeyi yönergeler verildikten sonra otomatik olarak kendi başına yapar.

6. Bilgisayarlar iki sayının karşılaştırılması gibi mantıksal işlemleride yaparlar.

Yukarıda verilen açıklamalardan sonra böyle özelliklere sahip bir makinayı bilgisayar olarak adlandırmak doğru olmasa gerek. Çünkü bu makinalar "bilgi" saymaz. Bir dereceye kadar "veri" sayar. Veri ve bilgi ise farklı kavramlardır. Bilgiye verilerin anlamlı biçime konulmuş sonucu ya da "işlenmiş veri", veriye ise "ham bilgi" diyebiliriz (Uman, 1973). Kimi zamanda bu makinaya İngilizcede verilen "data processing machine" adından dolayı "bilgi işlem makinası" denilmektedir. Halbuki "data" sözcüğü bilgi değil veriler demektir. Dolayısıyla, "veri işlem makinası" denilseydi belki daha doğru bir adlandırma olacaktı.

Diğer yandan, yine sık sık olmasa da, yukarıda özellikleri belirtilen makina elektronik beyin olarak adlandırılmaktadır. Bu makinanın elektronik olarak işleyişi, bir bellek sisteminin oluşu ve iki sayıyı karşılaştırma gibi özelliklerine ek olarak bir beyin gücü, söz konusu değildir. Çünkü beyin bir düşünme ve karar verme işlevi görmesine karşın bu makinaların böyle bir özelliği yoktur. Her ne kadar bilgisayar çeşitli koşulları karşılaştırır, olası hareket yönlerinden birisini verilen yönergelere göre seçmeye karar veriyorsada bu kendi başına verdiği bir karar değildir.

Tüm bunlara ek olarak makinanın İngilizce hesaplayıcı anlamına gelen "computer" adının da doğrudan dilimize "kompüter" olarak girdiği görülmektedir. Bu sözcüğün yabancılığı bir yana bırakılırsa, daha doğru bir adlandırmadır. Çünkü bu makinalar aslında bir hesap-

layıcıdır. Bu açıklamalara karşın dilimize giren ve yerleşen bir sözcük olduğu için bundan sonraki sayfalarda da elektronik bir hesaplayıcı ve veri işleyen bir makina demek olan "bilgisayar" sözcüğü kullanılacaktır.

1.2 Bilgisayarların Tarihsel Gelişimi

Günümüzde geniş bir kullanım alanı bulan bilgisayarların gelişimi oldukça uzun bir zaman almıştır. İlk sayısal hesaplama aracı olarak, Doğu'da geliştirilen ve daha sonraları Ortaçağ başlarında Batı'ya tanıtılan "abacus" gösterilebilir. Ancak blardo oyununda sayı saymak için kullanılan alete benzeyen abacus'ün bugünkü bilgisayarın başlangıcı olamayacağı açıktır. Bir aletin ya da aracın bilgiyasarların gelişiminin başlangıcı olarak gösterilebilmesi için bir dereceye kadar otomatik bir özelliğinin olması gerektiği düşünülürse 16. yüzyıldan başlamak gerekmektedir. 1642 yılında, büyük Fransız matematikçisi Blaise Pascal (ki bu tarihte 19 yaşındaydı) gümrük memuru olan babasına yardım etmek için toplama ve çıkarma işlemlerini yapan mekanik bir hesaplama makinası geliştirdi. Daha sonraları 17. yüzyılda Leibniz, Pascal'ın makinasını çarpma ve bölme işlemlerini de yapacak bir biçimde geliştirdi. Böylece bilgisayarların tarihi gelişim süreci içinde ilk kez olarak aritmetik işlemlerin mekanize edilebileceği fikri gerçekleşmiş oldu (Ralston, 1971).

Bilgisayarların gelişiminde ikinci büyük fikir, 19. yüzyılın başlarında İngiliz matematikçisi Charles Babbage tarafından ortaya atıldı. Endüstri Devriminin başlangıç yıllarında yaşamış olan Babbage yığın üretimde kullanılan makina tekniklerinin matematiksel tabloların hazırlanmasınada uyarlanabileceğini düşündü ve 1822'de "Fark motoru (difference engine)" adını verdiği bir hesaplayıcı geliştirdi. Daha sonraları 1833 yılında Babbage özel amaçlı ve logaritma gibi matematiksel tabloların hazırlanmasında kullanılan fark motorundan ayrı olarak genel amaçlı ve bugünkü bilgisayarların özelliklerine sahip otomatik bir hesaplayıcı geliştirmeyi düşündü.

"Analitik motor (analitical engine)" adını verdiği bu hesaplayıcının kuramsal düzeyde en büyük özelliği, programlanabilir ve otomatik olması yanında yönerge ve verilerin delikli kartlar aracılığı ile hesaplayıcıya verilmesi idi. Analitik motorun kapsamlı çizimleri ve hatta bazı parçalarının da imal edilmesine karşın 19. yüzyılın ortalarındaki teknoloji Babbage'ın bu fikirlerini gerçekleştirmesine engel oldu (Ralston, 1971).

Fakat bu alandaki çalışmalara paralel olarak, Amerika Nüfus Sayımı Bürosu elemanlarından Herman Hollerith 1890 nüfus sayımı verilerini çabuk işlemek için delikli kartların kullanıldığı bir tablo yapma makinası geliştirdi (Hopeman, 1971). Böylece veri işlemede delikli kartlardan yararlanma olanağı doğmuştu.

Bu gelişmelere karşın Babbage'ın analitik motorunun ilkelerini kapsayan bir hesaplayıcının gerçekleştirilmesi 1939-44 yılları arasında Howard Aiken tarafından Mark I'ın yapımına kadar sürdü. Elektronik bir hesaplayıcı olan Mark I veri işlemede programlı kontrola doğru atılan çok önemli bir adımdı. Hızı dakikada 200 işlem olan bu hesaplayıcı veri ve yönergeleri delikli kağıt şeritler aracılığı ile alıyordu.

Aynı dönemde J.P. Eckert ve J.W. Mauchly 1945 yılında ilk elektronik hesaplayıcı olan ENIAC'ı (Electronic Numerical Intergrator and Calculator) geliştirdiler (Hopeman, 1971). Bilim ve teknolojinin ilk modern sayısal bilgisayarı yapmaya doğru ilerlediği bu dönemde Eckert, Mauchly, John von Neumann ve arkadaşları elektronik hesaplayıcılara hangi aritmetik işlemi yapması gerektiğini belirten yönergelerin hesaplayıcıların içsel saklama sistemi adı verilen bir bölümünde depolanabileceği fikrini ileri sürdürler. Böyle bir bilgisayar geliştirmek için 1946 yılında Pennsylvania Üniversitesinde EDVAC'ın (Electronic Discrete Variable Automatic Computer) ve Institute For Advanced Study'de (IAS) von Neumann ve arkadaşları tarafından IAS bilgisayarının yapımına başlandı. Ancak her iki çalışmada 1952 yılına kadar bitirilemedi.

Bu sırada ilk modern sayısal bilgisayar 1949 yılında Cambridge Üniversitesinde M.V. Wilkes ve arkadaşları tarafından EDSAC (Electronic Delay Storage Automatic Calculator) adı altında imal edildi. Fakat ticari amaçla piyasaya sürülen ilk sayısal bilgisayar Eckert ve Mauchly tarafından Remington-Rand Şirketi için 1951 yılında yapılan UNIVAC-1 oldu (Ralston, 1971).

Daha sonraları IBM, RCA, Honeywell, Burroughs, v.b. firmalarında bilgisayar piyasasına girmesi ile artan rekabet sonucu, EDSAC ve UNIVAC-1'den günümüze kadar olan zaman içinde bilgisayarların gelişme bakımından dört kuşaktan geçtiği kabul edilir. Birinci kuşak bilgisayarlarda radyo lambası, İkinci kuşak bilgisayarlarda transistor, üçüncü kuşaklarda ise birleşik devre (integrated circuitry) kullanılmıştır. Dördüncü kuşak bilgisayarlar ise mikro bilgisayarlar olarak bilinirler.

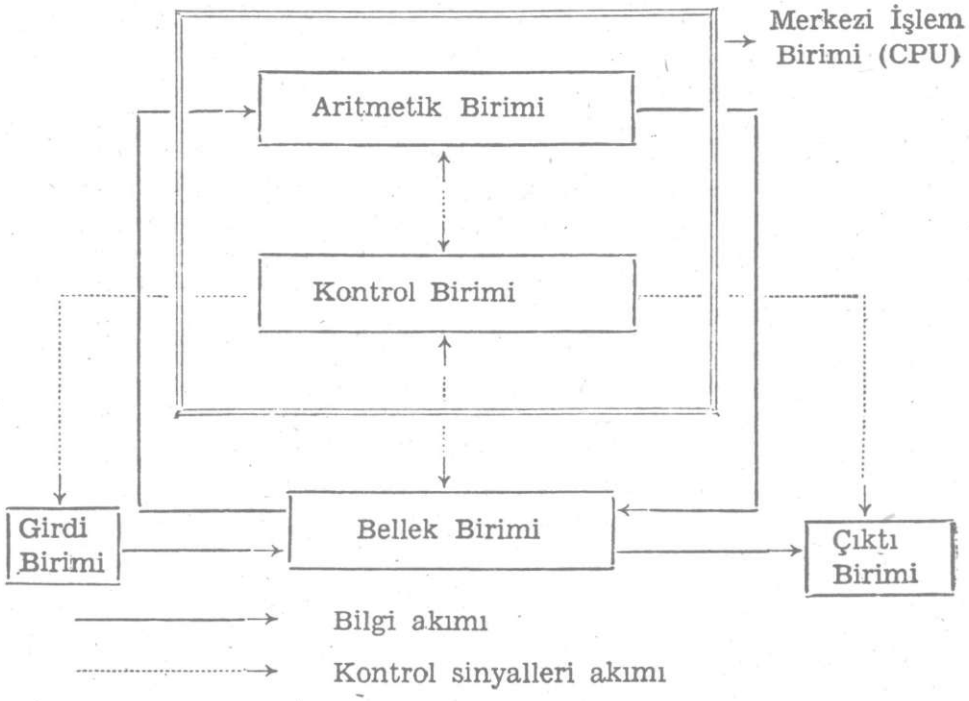
Birinci kuşak bilgisayarlara UNIVAC-1 ve IBM 650, ikinci kuşağa IBM 1401, 7094 ve 1620, üçüncü kuşağa IBM System/360 ve 370 örnek

olarak gösterilebilir. Dördüncü kuşaklara mikro kişisel bilgisayarlar verilebilir.

Bilgisayar teknolojisi alanındaki gelişmeler günümüzde de hızla devam etmekte ve bugün cepte taşınabilecek kadar küçük ve kullanıcının doğrudan iletişimde bulunabileceği özel bilgisayarlar konusunda çalışmalar yapılmaktadır. Ayrıca yapay uslu bilgisayarlar konusunda da çalışmalar hızlanmıştır.

1.3. Bilgisayarların yapısı, İşleyişi ve Kullanımı

Bilgisayar olarak adlandırılan elektronik hesaplama sistemi genel olarak beş önemli birimden oluşmaktadır. Şekil 1'de de gösterildiği gibi bu birimler : girdi, bellek, kontrol, aritmetik ve çıktı birimleridir.



Şekil 1 : Bilgisayar Sisteminin Yapısı

1. **Girdi Birimi** : İşlenecek verilerin ve bu verilerin nasıl işleneceğini belirten yönergelerin bilgisayara aktarılmasını sağlayan kısımdır. Aktarma işi için delikli kartlar, delikli kağıt şerit, magnetik şerit ve ekranlı terminaller (CRT) en çok kullanılan araçlardır. Giriş birimi olarak delikli kart ve şerit okuyucular, kurşun kalemle işaretlenmiş bil-

gileri okuyabilen "im" ya da işaret okuyucular (mark readers) ve ekranlı terminaller en çok kullanılanlardır. Ancak bunlar içinde de en yaygın olarak bilinen ekranlı terminal ve delikli kart okuyucudur. Bu okuyucular ortalama olarak dakikada 1500-2000 kadar satır ya da kart okumaktadırlar. Kartlar ve satırlar standart olup 80 sütun kapsar.

2. **Bellek** (hafıza) **Birimi** : Bilgisayara aktarılan veri ve yönergelerin saklandığı, yani depo edildiği birimdir. Veriler üzerinde yönergelere göre yapılacak işlemlerin sonuçlarıda yine bu birimde depolanır. Bellek sistemi, manyetik çekirdek denilen toplu iğne başı büyüklüğünde halkalardan oluşmaktadır. Bu halkalar içinden geçirilen elektrik akımına bağlı olarak manyetik çekirdekler sağa ya da sola doğru mıknatıslanmaktadır.

Mıknatıslanma yönüne bağlı olarak bilgileri depolamak olası olmaktadır. Örneğin, çekirdeğin mıknatıslanma yönü sol ise 1 sağ ise 0 diyelim. Eğer 1 ve 0 ile tüm sayıları, harfleri ve hesaplamalarda gereksinim duyulan diğer özel işaretleri gösterebilecek ikili bir belirleme sistemi kullanırsak istediğimiz bilgiyi bir çok manyetik çekirdek kullanarak tanımlayabiliriz. Örneğin, 1001 gibi ikili bir tanımlama 9 rakkamını göstermek için kodlanabilir. Böylece her rakkam, harf ve özel işarete bir kod verilerek istenilen bilgiler bellekte saklanabilir ve elektrik akımları ile istenildiği biçimde değiştirileblir. Görüldüğü gibi en basit anlamı ile bilgisayar bellek sistemi, bir elektrikle mıknatıslanma olayının ikili bir kodlama sistemi ile yorumlanmasıdır.

Manyetik çekirdeklerden oluşan belleğin hacmi yani depolayabileceği bilgi miktarı bilgisayar sistemine göre değişir. En büyük hacimli bellek sisteminin bile depolayabileceği bilgi miktarı sınırlıdır. Ancak manyetik şerit, disk ya da tambur adı verilen yardımcı ek bellek araçları kullanılarak ana bellek hacmini (kapasitesini) artırmak olasıdır. Çoğu bilgisayar sistemlerinde ek bellek kapasitesi ana bellek kapasitesini aşmaktadır.. Ancak yardımcı bellekte bilgi depolanması yine de ana bellek aracılığı ile olmaktadır. Genellikle bir bilgisayarın ana bellek kapasitesi K harfi ile belirtilir. $1\,K = 2^{10} = 1024$ sözcüğü temsil etmektedir. Sözcük bir yönerge, bir sayı ya da bir karakter dizisi olabilir. Örneğin, belleği 48 K olan bir bilgisayarın bellek kapasitesi $48 \times 1024 = 49152$ sözcüktür. Modern bilgisayarlarda ana bellek gücü 1 milyon sözcüğü geçmektedir.

3. **Kontrol Birimi** : Bir çok elektronik devrelerden oluşan bu birim bilgisayarın bütün etkinliklerini denetler: yönergeleri yorumlar ve hangi sıra düzeni içinde işleneceğini belirler, verileri bellekten alıp

aritmetik işlem için aritmetik birime gönderir ve aritmetik birimde işlenmiş olarak gelen verileri (bilgileri) bellekte depo eder.

4. **Aritmetik Birim** : Bu birim kontrol biriminden gelen sinyallere göre, bellek biriminden gelen verilere dört aritmetik işlemi uygular. Yani aritmetik işlemleri yapar. Bu birim ile kontrol birimi birlikte Merkezi İşlem Birimi (CPU) olarak adlandırılır.

5. **Çıktı Birimi** : Yapılan işlemlerin sonucunu bellek biriminden alır ve kullanıcıya verir. Sonuçları kartlara ya da şeritlere delen delgi araçları ile sonuçları bir ekranda gösteren göstericiler (ekranlı terminaller) bilinen çıktı birimleri olmasına karşın, en yaygın kullanılan çıktı birimi sonuçları bir kâğıt üzerine normal daktilo biçiminde yazan ve yazıcı (printer) adı verilen bir makinadır. Bilgisayarın kullanıcıya verdiği sonuçlar çoğu kez "çıktı" olarak adlandırılır. Bu sözcük bilgisayar işlemlerinin sonucu anlamında burada da kullanılacaktır. Çıktının yazıldığı kağıt genel olarak her satırı 132 karakterlik 60 satırdan oluşur.

Bundan önceki sayfalarda belirtilen temel birimlerden oluşan bilgisayar genellikle Tek Sıra (Batch) ve Zaman Bölüşümlü (Time Sharing) kullanım olmak üzere iki farklı biçimde kullanılır.

Tek sıra kullanımda bilgisayar kullanıcılarının verdiği işler tek tek sıra ile işlenir. Kuşkusuz, bilgisayar sistemi yöneticileri kendilerine göre bir öncelikli işlem sistemi de geliştirebilirler. Bu tür kullanımda kullanıcı bilgisayar merkezine gider, işini verir (programını okutur), işlem sırasını bekler ve sonra işlem sonucu çıktıyı alır. Tek sıra kullanımda bilgisayara veri ve yönergeler genellikle delikli kartlarla verilir. Büyük işler (programlar) için verimli olan bu yöntem işlerin küçük ve sonucun kısa bir sürede alınmak istendiği durumlarda verimli değildir.

Zaman bölüşümlü kullanımda bilgisayarın aynı anda birden fazla kullanıcı tarafından kullanımı olasıdır. Her kullanıcının bilgisayara telefon hatları ya da mikro devrelerle (micro circuits) bağlı daktilo terminalleri (typewriter terminal) ya da ekranlı terminaller aracılığı ile bilgisayarla iletişim olanağı vardır. Terminaller genellikle bilgisayar merkezi dışında ve uzaktadır. Terminaller hem girdi hemde çıktı aracıdır. Bu nedenle kullanıcının bilgisayar merkezine gitmesi gerekmez. Veri ve yönergeler terminalin üzerindeki daktilo ile yazılarak iletilir.

1.4 Bilgisayarda Bilginin Belirtilmesi

Bilgisayarlar içinde bilginin belirtilmesi yalnızca 1 ve 0 rakkamlarının kullanıldığı ikili bir sistem aracılığı ile olmaktadır. Böyle bir

sistemin kullanılma nedeni bir önceki bölümde ifade edilmişti. Buna göre, elektrik yönüne bağlı olarak manyetik çekirdeklerin mıknatıslanma yönü sol ve sağ yerine 1 ve 0 rakkamları kullanılarak bir kodlama sistemi geliştirilebilir. İstenilen bilgiler böyle bir kodlama sistemi ile bilgisayarda belirtilebilir.

İkili sistemin temeli olan 1 ve 0 rakkamlarının her birine İngilizce karşılıkları olan "Binary digit" sözcüklerinin ilk iki harfi (Bi) ve son harfi (t) nin bileşiminden oluşturulan "bit" adı verilmektedir. Bir karakterlik bilgiyi (bir harf, bir rakkam ya da bir özel işareti) belirtmek için, kullanılan kodlama sistemine göre 6 ya da 8 bit bir "byte" oluşturur. BCD (Binary Coded Decimal) kodlama sisteminde bir byte 6 bit'ten, EBCDIC (Extended Binary Coded Decimal Inter-Change Code) kodlama sisteminde ise bir byte 8 bit'ten oluşur. İki ya da dört byte ise bir sözcük oluşturur. Aynı biçimde sözcükler yönergeleri, yönergeler ise programları oluştururlar.

1.4.1 İkili Sayı Sistemi

Bilgisayarlarda, bildiğimiz normal 10 tabanlı sayıların ikili sistemle nasıl gösterilebileceğinin bu noktada açıklanması uygun olacaktır.

Bilgisayarda kullanılan ikili sayı sistemi 10 tabanlı sayı sistemine çok benzer. Bu benzerlik aşağıdaki örnekle açıklanabilir. Üssü sıfır olan sayıların 1'e eşit olduğu (Örneğin, $10^0=1$) anımsanırsa, 10 tabanlı bir sayı olan 1982 on tabanına göre şöylede yazılabilir :

$$(1982)_{10}=1\times10^3+9\times10^2+8\times10^1+2\times10^0$$

Aynı biçimde 1001 gibi ikili bir sayı da 2 tabanına göre şöyle yazılabilir :

$$(1001)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0$$

Buna göre bu ikili sayının 10 tabanına göre değeri,

$$(1001)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0$$
$$(1001)_2=8+0+0+1$$
$$(1001)_2=(9)_{10}$$

olacaktır.

10 tabanlı sayıya dönüştürmek için bir başka örnek olarak 1111 ikili sayısını alalım.

$$(1111)_2=1\times2^3+1\times2^2+1\times2^1+1\times2^0$$
$$(1111)_2=8+4+2+1$$
$$(1111)_2=(15)_{10}$$

Diğer bir örnek.

$(101010)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$(101010)_2 = 32 + 0 + 8 + 0 + 2 + 0$

$(101010)_2 = (42)_{10}$

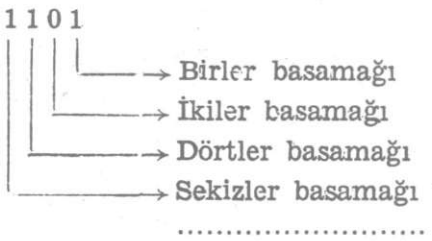
Yukarıdaki örneklerde görüldüğü gibi ikili sistemde 1 ve 0 bit'leri verilen herhangi bir ikili sayıda basamak sayısına göre 2 tabanı üslerinin varlığını (1) ve yokluğunu (0) göstermektedir.

Böylece, konu taban üssünün varlığı ve yokluğu olduğuna göre, 10 tabanlı sayıları 2 tabanlı (ikili) sayılara dönüştürmek için verilen tamsayı devamlı olarak 2'ye bölünür ve kalanlar yan yana yazılırsa ikili bir sayı elde edilir.

Örnek 1 : 13 sayısını ikili sayıya çevirelim.

13/2=6 Kalan=1 (sağdan ilkbasamak)
6/2=3 Kalan=0 (ikinci basamak)
3/2=1 Kalan=1 (üçüncü basamak)
1/2=0 Kalan=1 (dördüncü basamak)

Bulunan kalanları aşağıdan yukarıya doğru yanyana dizerek (1101) ikili sayısı elde edilir. Yani, $(13)_{10} = (1101)_2$ dir. Kuşkusuz kalanları aşağıdan yukarıya doğru yanyana dizmemizin nedeni; son basamak olan dördüncü karakterin ikili sistemin 1'ler, üçüncünün 2'ler, ikincinin 4'ler, ve birinci karakterinde ikili sistemin 8'ler basamağı olmasındandır. Yani ikili sistemde basamak düzeni aşağıdaki gibidir.



Anımsanacağı gibi onlu sistemde basamak düzeni : 1'ler, 10'lar, 100'ler, 1000'ler v.b. biçimini alır.

Örnek 2 : 42 sayısının ikili sisteme çevrilmesi.

42/2=21 Kalan=0
21/2=10 Kalan=1
10/2= 5 Kalan=0

5/2= 2 Kalan=1

2/2= 1 Kalan=0

1/2= 0 Kalan=1

Sonuç : $(42)_{10}=(101010)_2$

Onlu sayıların kesir kısımlarının ikili sisteme çevrilmesinde kesirli kısım 2 ile çarpılır, tamsayı kısmı çıkarılır ve kalan kesir 2 ile tekrar çarpılarak işlem aynı biçimde sürdürülür. Ancak çevirme işleminde kesir kısmı sıfırlanmadığı zaman yaklaşık bir değer alınır.

Örneğin : 0.59 kesrini ikili kesire çevirelim.

0.59×2=1.18 Çıkarılan tamsayı : 1

0.18×2=0.36 Çıkarılan tamsayı : 0

0.36×2=0.72 Çıkarılan tamsayı : 0

0.72×2=1.44 Çıkarılan tamsayı : 1

0.44 .....

.....

Elde edilen tamsayıları bu kez yukarıdan aşağıya doğru, yanyana dizerek (kesirli sayılar soldan sağa doğru $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots\ldots$ biçiminde 10 tabanlı sayılardaki 0.1, 0.001, 0.0001 ... gibi basamaklandırıldığı için) :

Sonuç : $(0.59)_{10}=(0.1001\ldots)_2$ olur.

Daha önce $(42)_{10}=(101010)_2$ olduğunu bulmuştuk.
Buna göre,

$(42.59)_{10}=(101010.1001\ldots)_2$ dir.

Kesirli ikili bir sayıyı 10 tabanlı kesirli sayıya çevirirken de kesirli kısım yukarıda olduğu gibi $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ ... gibi basamaklar biçiminde düşünülerek toplanır.

Örneğin : $(0.1101)_2$ sayısının onlu karşılığı,

$$(0.1101)_2=1\times\frac{1}{2^1}+1\times\frac{1}{2^2}+0\times\frac{1}{2^3}+1\times\frac{1}{2^4}$$

$$(0.1101)_2 = \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16} = \frac{13}{16} = 0.812$$

$$(0.1101)_2 = (0.812)_{10}$$

olur.

Başka bir örnek olarak (1101.0101) ikili sayısının onlu karşılığını bulalım :

$$(1101.0101)_2 = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times \frac{1}{2^1} + 1\times \frac{1}{2^2}$$

$$+0\times \frac{1}{2^3} + 1\times \frac{1}{2^4}$$

$$(1101.0101)_2 = 8+4+0+1+0+ \frac{1}{4} + 0 + \frac{1}{16}$$

$$(1101.0101)_2 = 13 + 0.312$$

$$(1101.0101)_2 = (13.312)_{10}$$

olur.

1.4.2 İkili Sayıların Dört İşlemi

1) Toplama : Bilindiği gibi 10 tabanlı sayılarda en büyük rakkam 9'dur. Toplama işleminde her basamaktaki toplam, 9'u geçtiği zaman toplamdan yeteri kadar taban 10 çıkarılır ve bitişik basamağa elde olarak aktarılır. Örneğin, 17+18 toplama işlemini yaparsak, birler basamağında 7+8=15 olarak bulunur. 15'den 1 tane 10 çıkarılır geriye kalan 5 kaydedilir. Çıkarılan 10 bitişik onlar basamağına elde 1 olarak aktarılıyor ve bu basamakta daha önce bulunan sayılara ekleniyor. Böylece 1+1+1=3 olarak kaydediliyor, yani 17+18=35 oluyor.

Yukarıda açıklanan 10'lu yöntem ikili sayıların toplanmasında da benzer biçimde izlenir. Ancak bu kez en büyük rakkam 1 ve tabanda 2'dir. Her basamakta toplam 1'den büyük olduğu zaman yeteri kadar 2 çıkarılır ve bir sonraki basamağa elde olarak aktarılır.

Örnek 1 :

13		1101
6		110
+ 11		+ 1011
30	işlemini ikili olarak yaparsak	11110 bulunur.

İki tabanlı işlemin yapılışını açıklayalım :

Birler basamağı : $1+0+1=2$ ve $2-2=0$ elde 1
İkiler basamağı : Elde $1+0+1+1=3$ ve $3-2=1$ elde 1
Dörtler basamağı : Elde $1+1+1+0=3$ ve $3-2=1$ elde 1
Sekizler basamağı : Elde $1+1+1=3$ ve $3-2=1$ elde 1
Onaltılar basamağı : Elde $1=1$ ve $1=1$ olur.

Sonuç : $(11110)_2$ dir.

Örnek 2 :

27		11011
30		11110
+ 12		+ 1100
69	ikili karşılığı	1000101

Yani, $(11011)_2+(11110)_2+(1100)_2=(1000101)_2$ olur.

2) Çıkarma : Onlu sistemin yöntemi aynı şekilde izlenir.

Örnek : 1

30		11110
− 27		− 11011
03	ikili karşılığı	00011

Açıklaması :

Birler basamağında : 0−1 mümkün olmadığı için ikiler basamağından 1 taşınır ve $2-1=1$ bulunur.

İkiler basamağında : Dörtler basamağından 1 taşınır ve $2-1=1$ bulunur.

Dörtler basamağında : Daha önce 1 alındığı için $0-0=0$ bulunur.

Sekizler basamağında : $1-1=0$ bulunur.

Onaltılar basamağında : $1-1=0$ bulunur.

Böylece bulunan değerlere göre sonuç : (00011) dir.

Örnek 2 :

69		1000101
− 12		− 1100
57	ikili karşılığı	0111001

3) Çarpma : Yöntemi onlu sistemin aynıdır.

Örnek 1 :

```
   14
 ×  3
 ----
   42
```

ikili karşılığı

```
   1110
 ×   11
 ------
   1110
+ 1110
 ------
 101010
```

Örnek 2 :

```
   27
 ×  5
 ----
  135
```

ikili karşılığı

```
   11011
 ×   101
 -------
   11011
  00000
+ 11011
 -------
 10000111
```

4) Bölme : Her iki sistemde yöntem tamamen aynıdır.

Örnek 1 :

```
   10 |2
 - 10 | 5
 ---- |--
   00 |
```

ikili karşılığı

```
   1010 | 10
 - 10   | 101
 ------ |
   0010 |
 -  10  |
 ------ |
   00   |
```

Örnek 2 :

```
   135 | 5
 - 10  | 27
 ----- |--
   035 |
 -  35 |
 ----- |
   00  |
```

ikili karşılığı

```
   10000111 | 101
 - 101      | 11011
 -------    |
   00110    |
 -  101     |
 -------    |
   00111    |
 -  101     |
 -------    |
    0101    |
 -  101     |
 -------    |
     000    |
```

Verilen örneklerinde belirttiği gibi ikili sayı sistemi ile dört işlem rahatlıkla yapılabilmekte ve sayılar basit bir biçimde gösterilmektedir.

II.

FORTRAN IV DİLİ İLE PROGRAMLAMA

2.1. Bilgisayar Programlarının Hazırlanması (Algoritma Geliştirme)

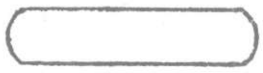
Bilgisayara verilen verilerin ne şekilde işleneceğini belirten yönergelerin tümüne program, yönergelerin de hazırlanmasına programlama denir. Bir bilgisayar programının hazırlanması, birbirlerinden kesin hatlarla ayrılmış olmamasına karşın, genellikle dört belirgin aşama izler.

1. **Problemin Çözümlenmesi**: Bu aşamada, belirlenen ve çözülmesi arzu edilen sorunla ilgili olarak ne tür verilerin ve sonuç olarak nelerin istendiği saptanır. Başka bir anlatımla, problemin çözümünde ne veriliyor? Ne isteniyor? Sorularına yanıt aranır.

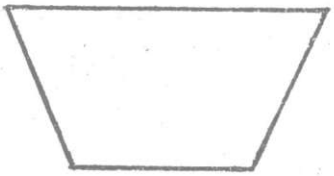
2. **Çözüm Yönteminin Bulunması**: Bu ikinci aşamada istenilen, sonuçların elde edilmesi için problemin nasıl çözüleceği belirlenir. Çözüm yöntemi belirlenirken bu arada problemin bilgisayarla çözülüp çözülmeyeceği de araştırılır. Çünkü her sorun bilgisayar ile çözülemez.

3. **Akış Şemasının Hazırlanması**: Belirlenmiş olan çözüm yönteminin bilgisayarda nasıl bir işlem sırası izleyeceği şekillerle gösterilir. Akış-şeması (flow-chart) çözüm yöntemini açıklığa kavuşturacağı gibi programı yazma aşamasında gözden kaçabilecek bazı işlemlerin önceden saptanmasını sağlar. Problemin çözüm yönteminin araştırılıp bulunması ve bulunan yöntemin bilgisayarda izleyeceği işlem sırası ile birlikte belirtilmesine "algoritma (algorithm)" adı da verirler. Böylece algoritma şimdiye kadar izlenen aşamaları kapsayan bir terim olmaktadır.

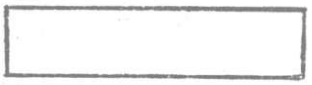
Akış-şemasının hazırlanmasında sık sık kullanılan bazı işaretler ve anlamları aşağıda verilmiştir.



Programda başlama, durma ve bitiş noktalarını gösterir.

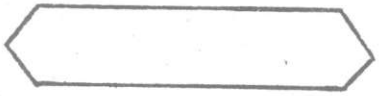


Girdi ve çıktı noktalarını belirtir.



Bir işlem yapıldığını belirtir. Yapılan işlem dörtgen içinde gösterilir.

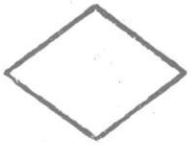
Ok işareti programın akış yönünü gösterir.



Döngü ya da altprogram bölümlerini gösterir.

Program bölümleri arasında bağlantı noktalarını belirtir.

Programın değişik sayfalardaki bölümleri arasında bağlantıyı belirtir.



Programda karar noktalarını gösterir.

4. **Programın Bilgisayar Dili ile Yazılması**: Program hazırlamanın bu son aşamasında, akış şemasına dayalı olarak çözüm yöntemi bilgisayar programlama dillerinden birisi ile kurallarına uygun ola-

rak yazılır. Bu amaçla, bilgisayarla kullanıcı arasında iletişim sağlamak için FORTRAN (FORmula TRANslation), ALGOL (ALGOrithmic Language), PL/1 (Programming Language/1), COBOL (COmmon Business Oriented Language), BASIC (Beginner's All-purpose Symbolic Instruction Code) gibi genel amaçlı programlama dilleri geliştirilmiştir.

Programlama dillerinden birisi ile yazılan programa "kaynak program (source program)" adı verilir. Kaynak program bilgisayarın anladığı kendi makina diline (machine-language) bir derleyici (compiler) aracılığı ile dönüştürülür ki, buna da "derlenmiş" yada "amaç program (object program)" denir. Derleyici bu işlevi "birleştirici dil (assembly language)" ile yapar.

Çalışmamızın bundan sonraki bölümlerinde, kaynak program hazırlamada çok yaygın olarak kullanılan ve yüksek düzeyde genel amaçlı bir programlama dili olan FORTRAN IV bilgisayar dilinin genel kuralları ve özellikleri işletmecilik alanındaki örneklerle açıklanmaya çalışılacaktır.

2.2 FORTRAN IV Yönergeleri ve Kodlaması

Yukarıda belirtildiği gibi bir program düzenli bir dizi yönergelerden oluşur. FORTRAN IV bilgisayar dili ile yazılmış bir programdaki yönergeler işlenebilir (executable) ve işlenemez (non-executable) diye iki kümeye ayrılır. İşlenebilir yönergeler programdaki sıralarına göre mantıksal bir düzende bilgisayar tarafından işlenir. İşlenemez yönergeler ise tanımlayıcı bir işlev görürler ve bilgisayar tarafından bir işlem yapılmasını gerektirmezler. FORTRAN programını oluşturan yönergeler çok çeşitli olmakla birlikte genel olarak amaçları bakımından: girdi-çıktı, atama ve denetim yönergeleri en sık kullanılan işlenebilir yönergelerdir. Bunlara ek olarak tanımlayıcı işlev gören ve bilgisayara programda kullanılan değişkenlerin türleri, bellek sisteminde yer dağılımları ile girdi ve çıktı işlemlerinin içeriğini belirten işlenemez yönergelerde aynı ölçüde programlamada sık sık kullanılan diğer yönergelerdir.

Bundan sonraki bölümlerde ayrı ayrı ele alacağımız FORTRAN IV yönergelerinin açıklamasına geçmeden önce, şu ana kadar kullandığımız "yönerge" sözcüğünün anlamı ile ilgili bir noktayı açıklığa kavuşturmak gerekir. Bilgisayarda verilerin nasıl işleneceğini belirten bir ifade (statement) olarak "yönerge" sözcüğü, bir "talimatı" ya da bir "komutu" belirtmektedir ve İngilizce bilgisayar programlanması yazınında kullanılan "statement" teriminin karşılığı olarak kullanılmıştır. Ancak Türkçe bilgisayar programlaması yazınında "statement" te-

riminin karşılığı olarak "yönerge" yerine "deyim" sözcüğü kullanılmaktadır. Deyim sözcüğü bilgisayarın ne yapması gerektiğini belirten bir ifadeyi okuyucuya anımsatmamasına karşın, yazınla terim birliği sağlamak için şimdiye kadar kullanılan yönerge sözcüğü yerine bundan sonra deyim sözcüğü kullanılacaktır.

O halde yinelersek, bilgisayara ne yapması gerektiğini bildiren düzenli bir dizi deyimden oluşan FORTRAN programı kaynak bir program olup birleştirici dil ile yazılmış derleyiciye aktarılır. Aktarma işi için çeşitli araçlar kullanılmakla beraber, en sık başvurulan yöntem kaynak programı ekranlı terminallerde ekranda yazmak ya da eskiden olduğu gibi IBM kartları da denilen standart kartlar üzerine delgi makinaları ile delerek vermektir (okutmaktır).

Böylece, FORTRAN kurallarına göre hazırlanmış ve derleme işi için ekranlı terminalde yazılmış ya da standart kartlar üzerine delinmiş bir rogram bilgisayarda derlenme ve sonra işlenme aşamasına gelmiştir denilebilir. Ancak program, yazım hatalarını görebilmek ve yazım işinin daha hatasız yapılabilmesini sağlamak için "kodlama" formları adı verilen kağıtlara yazılır. Programın ekranda yazıldığı satırlar ya da delindiği kartlar 80 sütunla standartlaştırıldıkları için kodlama formundaki her satır 80 sütunluk bir satırı ya da kartı temsil eder. Şekil 2'de bir ekranlı terminal ve Şekil 3'de bir kart ve kodlama formu gösterilmiştir.

FORTRAN IV dilinin yazılış kurallarını açıklamada kolaylık sağlamak için tüm açıklamalar delikli kartlara göre yapılmıştır. **Her kart ekranlı terminallerde bir satırı temsil eder.** Böylece bir karttaki bir sütun, ekranda bir karakter (harf ya da sayı v.b.) yazmak için gerekli yere karşılık olarak alınmalıdır. Örneğin, **bir kartta beş sütun** ekranda bir satırda **beş karakterlik** yer demektir. Aynı şekilde, **beş kart** ekranda **beş satır** demektir. 1970'li yıllarda daha çok kullanılan delikli kartlar günümüzde ancak üniversitelerde eğitim amacı ile kullanılmaktadır. Ancak yazılı anlatımda kolaylık sağladığı için ve ayrıca bu kitabın birinci baskısındaki ifadeleri fazla değiştirmemek için bu ikinci baskıda da delikli kartlara dayalı anlatım tercih edilmiştir.

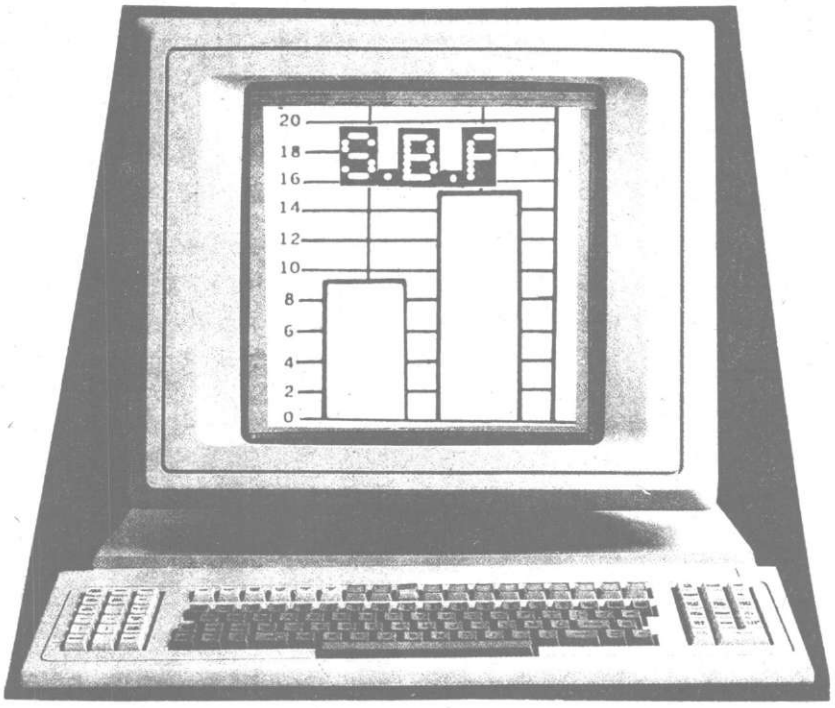
Bu açıklamalar ışığında hazırlanacak bir programı, ekranlı terminallere yazmada ya da delikli kartlara delmede yardımcı olma amacı ile kullanılan kodlama formunun her satırındaki 80 sütun rogramı yazmak için 5 bölgeye ayrılır.

1. **Bölge :** Bu bölge yalnız birinci sütunu kapsar ve ileti (mesaj) bölgesi ya da sütunu denilir. Programın anlaşılabilirliğini artırmak için programcı istediği iletileri programda belirtebilir. Bunun için

kartın 1. sütununa bir C harfi (Comment : yorum anlamında) konur ve ileti ondan sonraki sütunlara yazılır. Kimi FORTRAN derleyicileri C harfinden sonra bir boş sütun bırakılmasını ve iletinin ondan sonra yazılmasını ister. İletiler işleme konulmaz.

2. **Bölge** : 1'den 5'e kadar (beş dahil) olan sütunları kapsar ve deyim numaralama bölgesi adını alır. Programın işleyişi sırasında kimi deyimlere tekrar dönülmesi gereken durumlarda dönüş yapılacak olan deyimin belirtilmesi, numaralandırılması gerekir. Deyim numaralarının yazıldığı bu bölge en fazla 5 sütunlu olduğu için deyim numarası da 5 basamaklı bir tamsayıyı geçemez. Bir deyime verilen bir numara başka bir deyime verilemez. Yani numaralandırılacak deyimler farklı numara almalıdırlar.

3. **Bölge** : Bu bölge yalnızca 6. sütunu kapsar ve devam sütunu ya da bölgesi adını alır.



Şekil 2 : Ekranlı Bir Terminal

M.E.T.U

DEPT. OF COMPUTING SCIENCES

CENTRAL COMPUTING LABORATORY

FORTRAN CODING FORM

JOB NO.:

Program Örnek Form

Coded By

Checked By

Date in

Page of

C FOR COMMENT

STATEMENT NUMBER 1–5	CONT. 6	7–72	IDENTIFICATION 73–80
C ÖRNEK KODLAMA FORMU			1. Bilge
	*		Devam
	1		Sütunu
20	READ(5,2)A,B,C		Yönergeler
2	FORMAT(3X,3F5.2)		
64224	6.209364 2 0		Veriler
C ÖRNEK KART			

Şekil 3 : Örnek Bir Kodlama Formu ve Bir Veri Delme Kartı

4. **Bölge** : 7'den 72. sütuna kadar (72 dahil) olan sütunları içerir ve FORTRAN deyimlerinin yazıldığı bölgedir. Eğer bir deyim 7-72 sütunları arasına sığmayacak kadar uzun ise ya da programcı deyimin tümünü aynı satırda (kartta) yazmak istemiyorsa, 6. sütuna sıfırdan (0) farklı bir karakter (harf, rakkam, işaret) konularak deyimin istenilen geri kısmı bitişik kodlama satırına (yani karta) yazılabilir. Kimi derleyiciler arka arkaya birden fazla devam kartı kullanmaya izin vermezlerse de, çoğu derleyiciler 4 ya da 9 kimileri de peş peşe 19 devam kartı (arka arkaya dizilmiş ve bir tek deyimi belirten kartlar) kullanma olanağı verirler. Bir deyimi bitişik satırda devam ettirme olanağından dolayı çoğu kodlama formlarının sütun sayısı 80'den azdır.

5. **Bölge** : 72'den 80'e kadar olan sütunları kapsar ve belirtme bölgesi adını alır. Bu bölgeye programcı istediği herhangi bir bilgiyi yazabilir. Çünkü bu sütunlar program işleme sırasında göz önüne alınmazlar.

Yukarda belirtilen kodlama bölgelerine programcı tarafından uymak zorunludur. Ancak programda işlenecek veriler tüm 80 sütunda yazılabilir. Yani veriler için bölgeleme kuralları söz konusu değildir. Kodlama formu belirtilen kurallar çerçevesinde yazıldıktan sonra formun her satırı formdaki biçimi ile ekranlı terminale yazdırılır ya da her satır bir kartta olmak üzere yazılan program kartlara delgi makinalarında delinir.

2.3. Sabitler ve Değişkenler

Bir bilgisayar programı incelendiğinde bir takım sayıların ve alfabe harfleri ile başlayan sembolik isimlerin bulunduğu görülür. Bunlardan, programın bilgisayardaki işlenmesi sırasında değeri değişmeyen büyüklüklere sabitler, değeri değişen sembolik isimlere de değişkenler denir.

Bir sayısal değeri olan FORTRAN sabitleri, tamsayı sabitleri ve gerçel sabitler diye ilk aşamada ikiye ayrılır.

1. **Tamsayı** (integer) **Sabitleri** : Bu tür sabitler ondalık kesir noktası olmayan tamsayıları temsil ederler. Eğer sabit pozitif bir değer taşıyorsa önüne artı (+) işareti konulmayabilir. Fakat negatif değerli bir sayı ise eksi (—) işareti taşıması gerekir. Tamyasının büyüklüğü (IBM System/360'a göre) $2^{31} - 1 = 2147483647$ değerini geçemez. En küçük sayı değeri ise $-2^{31} = -2147483648$ dir.

Geçerli tamsayı sabitleri için aşağıdaki örnekler verilebilir.

— 1007	0
+ 400	526
42	999999

Aşağıdaki örnekler ise geçerli olmayan tamsayı sabitleridir.

42.	Nokta kullanılmaz
400,5	Virgül ve kesir kısmı bulunmaz
36240678594	Fazla büyük

2. **Gerçel (real) Sabitler**: Bu sabitler kesirli olsun ya da olmasın devamlı ondalık kesir noktası kullanılarak yazılan sayılardır. Gerçel sayılar ondalıklı ve üslü olmak üzere iki biçimde yazılır.

Ondalık yazma biçimine örnek olarak :

178.56	0.
— 15.0	0.0078
7.	— 3.

verilebilir.

Üslü yazma biçiminde ise gerçel sayı : bir mantis, üssü belirten bir E harfi ve üssün gücünü belirten en fazla iki basamaklı bir tamsayıdan oluşur. Mantis ondalıklı gerçel ya da tamsayılı biçimde yazılmış olan sayı kısmıdır. Bu tür sayı yazılımına örnek olarak :

17.85E +01	0.007E+4
17856E —2	0784E—03
— 1.5E1	—69.72E—14

gösterilebilir.

Örneklerde görüldüğü gibi E harfinden önce gelen sayılar (gerçel ya da tamsayı) mantis, E harfinden sonra gelen tamsayılar ise mantisin çarpılacağı üssün gücüdür. E harfi ise 10 tabanını gösterir. Böylece yukarıdaki örneklerin sayısal değeri :

$$17.856E+01 = 17.856 \times 10^{1} = 178.56$$
$$17856E-2 = 17856 \times 10^{-2} = 178.56$$
$$-1.5E1 = -1.5 \times 10^{1} = -15.$$
$$0.007E+4 = 0.007 \times 10^{4} = 70.$$
$$.784E-03 = 0.784 \times 10^{-3} = 0.000784$$
$$-69.72E-14 = -69.72 \times 10^{-14} =$$

Başka bir anlatımla E harfi üssün gücüne ve işaretine göre mantiste noktayı sağa ve sola hareketlendirmektedir. Örneğin, eğer üssün işareti (+) ve gücüde 1 ise mantisteki nokta bir basamak sağa kaydırılmakta, eğer üssün işareti (—) ve gücüde 3 ise mantisteki ondalıklı nokta 3 basamak sola kaydırılmaktadır. Eğer mantis bir tamsayı ise yani noktasız yazılmış ise bu durumda nokta mantisin en sağında varsayılır. Örneğin, 17856E — 2 üslü sayısında mantis 17856 tamsayısıdır ve 17856. gibi varsayılır. E harfini izleyen üssün işareti yazılmamış ise üs pozitif (+) olarak alınır. Üssün pozitif olduğu durumlarda işareti yazmak zorunlu değildir. Ancak negatif (—) ise mutlaka yazılmalıdır.

Üssün gücü iki basamaktan fazla basamağa yazılamaz. Ancak iki basamak kullanmak zorunluluğu da yoktur. Örneğin, gücü 3 olan bir üs işaretten sonra 03 biçiminde yazılacağı gibi yalnızca 3 olarakda yazılabilir.

Doğru yazılmış gerçel sayılara örnekler :

0.	191.1E—10
142E+03	—30.
0.0	.7
734.644	42.0
—513.45E—5	6.0E+02
0.000647	6.E3

Bir gerçel sayının alabileceği en küçük ve en büyük değerler IBM System/360 göre 10^{-78} ile 10^{+75} arasında sınırlandırılmıştır. Öte yandan normalleştirilmiş biçimlerde mantis 7 basamaklı bir sayıdan büyük olamaz. Örneğin, 86945.23 sayısı çeşitli biçimlerde üslü olarak yazılabilir : 869.4523E+2, 8694.523E1, 8.694523E+4, .8694523E5 gibi. Bunlardan en sonuncu biçim (.8694523E5) yani ondalık noktanın sıfır olmayan ilk basamaktan önce geldiği biçim, normalleştirilmiş biçimdir. Bu durumlarda mantis 7, kimi durumlarda 8 basamağı geçemez.

Eğer mantis 8 ya da daha fazla basamaklı ise sabit çift hassaslıklı (double precision) gerçel sabit adını alır. Başka bir ifade ile gerçel sabitler hangi biçimde yazılmış olursa olsun eğer basamak sayısı 8-16 arasında ise sabit çift hassaslıklı bir sabit olarak alınır. Eğer basamak sayısı 8 den az ise tek tassaslıklı gerçel sabittir. Çift hassaslıklı sabitler üslü olarak belirtildiği zaman üs E yerine D ile gösterilir.

Örneğin;

78.43264837
862.3D+03
76485.4382E—2

sabitleri çift hassaslıklı gerçel sabitlerdir. 862.3D+03 sayısının değeri E ile olduğu gibi 862300. olarak alınır. Burada üs D ile gösterildiği için sabit çift hassaslıklı olarak değerlendirilir. 76485.4382E—2 sabiti E ile yazılmış olmasına karşın mantisin basamak sayısı fazla olduğu için çift hassaslıklı olarak alınmaktadır. Böyle durumlarda üssün E ya da D ile belirtilmesi durumu değiştirmez. Çift hassaslıklı sayılar için bilgisayar bellek sisteminde iki depolama bölgesi kullanılır. Halbuki tek hassaslıklılar için bir tek depolama bölgesi ayrılır.

Hatalı olarak yazılmış gerçel sayılara örnek.

3.1E	Üs yazılmamış
830	Ondalıklı nokta kullanılmamış.
14,50	Virgül kullanılmaz. Nokta gerekli.
85D99	10^{75}'den büyük
6.E—83	10^{-78}'küçük

Değişkenler : Bilgisayar programının işlenmesi sırasında değeri değişen ve sembolik isimlerle gösterilen büyüklüklere değişken denilir. Sabitlerin değişmeyen değerlerine karşılık değişkenler programın işlenmesi süresince değişik zamanlarda ve noktalarda farklı değer alabilir.

Bir FORTRAN değişkenini sembolik isimlerle belirtmek için aşağıdaki kurallar izlenmelidir.

1. Yalnızca alfabetik ve nümerik karakterler kullanılmalıdır.

2. Sembolik isim altı karakterden fazla olmamalıdır.

3. Sembolik ismin ilk karakteri bir alfabetik karekter (alfabe harfi) olmalıdır.

Bu kurallara göre oluşturulan sembolik isimlerin ya da değişkenlerin çeşitli biçimlerde sınıflandırılmalarına karşın burada bir önceki konuya ilişkin olarak yalnızca tamsayı ve gerçel değişken türleri açıklanacaktır. Diğerleri bundan sonraki bölümlerde yeri geldikçe belirtilecektir.

1. **Tamsayı** (Integer) **Değişkenler** : Bu değişkenler biraz önce açıklanan kurallar çerçevesinde I harfinden N harfine kadar, yani I,

J, K, L, M, N harflerinden biri ile başlayan ve aldıkları değer bir tamsayı olan değişkenlerdir.

Doğru yazılmış bazı tamsayı değişkenlerine örnek :

IYIL
N
MEMURL
L843A
NIMALY
J64

2. **Gerçel (Real) Değişkenler :** Bu değişkenlerde yukarıda belirtilen harfler (I,J,K,L,M,N) dışında kalan harflerden birisi ile başlayan ve gerçel bir sayı değeri alan değişkenlerdir. Bu tür değişkenlere örnek olarak aşağıdaki örnekler verilebilir :

SERMAY
GSMH
ZARAR
GELIR
A41
X07B
Y
QKAR

Programcı programın anlaşılabilirliğini artırmak için değişken isimlerini belirtilen kurallar çerçevesinde istediği biçimde oluşturabilir.

Aşağıda verilen örnekler geçerli olmayan FORTRAN değişkenleridir.

SERMAYE	Altı karakterden fazla.
1981K	İlk karakter bir harf değil.
I/0	Özel karakter (/) kullanılmaz.
GIDER.	Alfanümerik özel karakter (.) kullanılmaz.
AD 3	Karakterler arasında boşluk (aralık) bırakılmaz.

2.4. Aritmatik İşlemler

Bir aritmetik işlem sabitler ve/veya değişkenler arasındaki ilişkileri belirten bir ifadedir. Örneğin A+B, 4/8, GELIR—GIDER gibi. İliş-

kinin nasıl olduğunu ise işlem belirleyicisi göstermektedir. Aritmetik işlem belirleyicileri aşağıdaki sembollerle gösterilir :

Toplama	+
Çıkarma	—
Çarpma	★
Bölme	/
Üs alma	★★

Örnek :

Cebirsel ifade	FORTRAN karşılığı
A+4.2—B	A+4.2—B
A . Z—4.0	A★Z—4.0
$(K/4)^3$	(K/4)★★3
$B^2.D$	B★★2★D
$\frac{A+B}{C^4}$	(A+B)/C★★4

FORTRAN aritmetik işlemleri aşağıda belirtilen kurallar çerçevesinde yapılır.

1. İşlemler belirli bir hiyerarşik düzen içinde sıra ile yapılır. Buna göre .

a) Aritmetik ifadeler soldan sağa doğru değerlendirilir.

b) İlk önce üs alma işlemleri yapılır.

c) İkinci olarak çarpma ve bölme işlemleri yapılır.

d) Son olarak da toplama ve çıkarma işlemleri yapılır.

Bu sıra düzeni birbirini izleyen aritmetik işlemlerden hangisinin öncelikle yapılacağını belirtir. Ancak, yukarıdaki hiyerarşi kuralına göre IBM System/360'da, eğer birinci işlem belirleyicisi ikinci belirleyiciye eşit ya da daha büyük bir hiyerarşik düzeyde ise birinci işlem yapılır. Eğer ikinci işlem belirleyicisi birinciden daha üst hiyerarşik düzeyde ise, ikinci belirleyici bu kez üçüncü belirleyici ile karşılaştırılır. İkinci üçüncüye eşit ya da büyükse ikinci işlem yapılır. Değilse üçüncü belirleyici ile dördüncü aritmetik işlem belirleyicisi karşılaştırılır ve işlemler bu düzen içinde devam eder. En son işleme gelindiğinde geri kalan işlemler (atlanılanlar) sağdan sola doğru yapılır (ters sıra). Örnek olarak aşağıda verilen işlem aşamalı olarak işlenmiştir.

4.0+3.0★5.0−16./2.0 ★★2

4.0+15.0−16./2.0★★2 ya da 4.0+3.0★5.0−16./4.0

19.0−16./2.0★★2 ya da 4.0+15.0−16./4.0

19.0−16./4.0 » » 4.0+15.0−4.0

19.0−4.0 » » 19.0−4.0

15.0 » » 15.0

bulunur.

2. İstenilen işlemlere öncelik vermek için ayraç kullanılabilir. Ayraç içindeki işlemler öncelikle yapılır. Eğer ayraçlar içiçe ise en içteki ayraçlardan başlanıp en dıştakine doğru gidilerek ayraç içindeki işlemler yapılır. Örneğin,

$$\frac{a+b}{c+d}$$

cebirsel ifadenin FORTRAN karşılığı A+B/C+D değildir. Eğer böyle yazılırsa istenilen işlem

$$a+\frac{c}{b}+d$$

biçiminde değerlendirilecektir. Bu durumu önlemek için ayraç kullanmak gerekecektir. Ayraç kullanımı ile,

$$\frac{a+b}{c+d}$$

ifadesinin FORTRAN karşılığı (A+B) / (C+D) biçiminde yazılmış olacaktır. Böylece ayraçlar ayrı ayrı değerlendirilir ve işlem yapılır.

Örnek : 8.0+7.0/(4.0+(6.0/2.0)) işleminin sonucu

8.0+7.0/(4.0+3.0)

8.0+7.0/7.0

8.0+1.0

9.0

olarak bulunur.

Örnektede görüldüğü gibi açılan her ayracı kapayan bir eş ayracın tamamlaması gerekir. Yani açılan ve kapanan ayraçlar sayıca birbirine eşit olmalıdır.

Örneğin;

$$\frac{X}{Y.Z}$$

ifadesi FORTRAN karşılığı ile X/(Y★Z) ya da X/Y/Z olarak yazılabilir. Çünkü son ifadesinin belirttiği : X değeri Y'ye bölünecek çıkan sonuçta tekrar Z'ye bölünecektir. Cebirsel olarak bu da,

$$\frac{\frac{X}{Y}}{Z} = \frac{X}{Y.Z}$$ demektir.

Burada belirtilmek istenen nokta, ayraçların kolaylık sağladığı ve kullanılmaktan çekinilmemesidir.

3. Çoğu FORTRAN IV derleyicileri bir aritmetik ifadedeki sabit ya da değişkenlerin aynı türden olmasını gerektirir. Bu kurala göre aşağıdaki aritmetik işlem yapılamaz.

GELIR — MALYET

Çünkü GELIR değişkeni gerçel MALYET ise bir tamsayı değişkenidir. Ancak aşağıda belirtildiği gibi MALYET değişkeni gerçel biçime dönüştürülebilir.

REAL MALYET

Böylece MALYET değişkeni gerçel (real) bir değişken olarak işleme konulur. Benzer şekilde eğer işlem tamsayılı olarak istenilirse GELIR değişkeni,

INTEGER GELIR

ifadesi ile tamsayı biçimine dönüştürülebilir. Görüldüğü gibi tamsayı değişkeninin önüne REAL yazmakla değişken gerçel olarak, gerçel değişkenlerin önüne de INTERGER yazılarak değişkenlerin tamsayı değişkeni olarak işleme konulması sağlanmaktadır.

REAL ve INTEGER sözcükleri değişkenlerin türünü ya da tipini derleyiciye bildirdikleri için "tip bildirme" deyimleri olarak adlandırılırlar. Bu deyimler aracılığı ile ayrı türden olan değişkenler aynı türe dönüştürülür. Kuşkusuz, eğer programcı değişkenleri oluştururken aynı türden yaparsa bu tip bildirme deyimlerine gerek yoktur.

Birden fazla değişkenin birtek tip bildirme deyimi ile türü belirtilebilir. Bunun için değişkenlerin birbirinden virgül ile ayrılması yeterlidir. Örneğin;

REAL KAR, ISCI, N

INTEGER YIL, SAYI

ifadeleri ile KAR, ISCI ve N tamsayı değişkenlerinin gerçel olarak, YIL ve SAYI gerçel değişkenlerinin de tamsayı değişkenleri olarak işleme konulması sağlanmaktadır.

Aritmetik işlemlerle ilgili bir başka tip bildirme deyimi çift hassaslıkla (double precision) ilgilidir. Eğer hassaslığı artırmak için bir değişkeni çift hassaslıklı yapmak istersek DOUBLE PRECISION tip bildirme deyimi kullanılır. Örneğin A,B SAL değişkenlerinin değerlerini çift hassaslıklı istersek;

DOUBLE PRECISION A,B, SAL

ifadesi yeterlidir. Böylece üç değişkenin her biri için iki depolama bölgesinin ayrılması bildirilmektedir.

4. Tamsayılı artimetik işlemlerde sonuçlar tamsayı olarak bulunur. Bu nedenle elde edilen sonucun kesir kısmı ne olursa olsun atılır.

Örnek 1 :

9/5 * 2 **3+14	İşlemin sonucu :
1 * 2 ** 3+14	(9/5=1.8 işleminin sonucu 1 olarak alınır)
1*8+14	
8+14	
22	olarak hesaplanır.

Örnek 2 :

3/6 * (4 ** 2+28/4+20)		İşlemin sonucu,
3/6 * (16+28/4+20)		
3/6 * (16+7+20)		
3/6 * 43		
0 * 43		(3/6 tamsayı işlemi sonucu 0.5' in kesir kısmı atılır ve 0 olarak değerlendirilir)
0	bulunur.	

Yukarıdaki örneklerde görüldüğü gibi tamsayılı aritmetik işlemlerde dikkatli olmak gerekir. Eğer zorunluluk yoksa bu tür aritmetik işlemlerden sakınmak en iyi yoldur.

Diğer yandan aritmetik işlemdeki sabit ya da değişkenlerden bir tanesi gerçel ise işlem gerçel biçimde yapılır. Böyle işlemlerde tamsayı sabiti ya da değişkeninin değeri gerçel biçime dönüştürülür ve işlem ondan sonra yapılır. Bu tür işlemler biraz önce açıklanan kural 3 dışında kalan ve karışık işlem kabul eden IBM System/360 gibi FORTRAN derleyicileri için sözkonusudur.

Örnek 1 :

6/10 ★ 3.0+2/4.0 ★ 3	karışık işlemin sonucu :
0 ★ 3.0+2/4.0 ★ 3	(6/10 tamsayılı işlemin sonucu 0'dır)
0.0 ★ 3.0+2/4.0 ★ 3	(0 tamsayısı 0.0 gerçel sayıya dönüştürülür).
0.0+2/4.0 ★ 3	
0.0+2.0/4.0 ★ 3	(2 tamsayısı gerçel 2.0 olur)
0.0+0.5 ★ 3	
0.0+0.5 ★ 3.0	(3 tamsayısı 3.0 yapılır)
0.0+1.5	
1.5	bulunur.

Örnek 2 :

4.0 ★ 3.0+5/3	
12.0+5/3	
12.0+1	(5/3 tamsayısı işlemi sonucu 1'dir)
12.0+1.0	(1 tamsayısı 1.0 gerçel yapılır)
13.0	bulunur.

Örneklerdende görlüdüğü gibi karışık işlemlerde dönüştürme işlemi nedeni ile işlem aşamaları uzamaktadır. Bu nedenle bu tür işlemlerden sakınmak gerekir. Yani işlemdeki değerlerin aynı türden olmaları tercih edilmelidir.

5. İki aritmetik işlem belirleyicisi hiçbir zaman yanyana yazılamaz. Bu kurala göre X★−Y gibi bir işlem geçerli değildir. −Y ★X ya da X★(−Y) olarak yazılmalıdır.

6. Gerçel bir sabit gerçel ya da tamsayılı bir üsse yükseltilebilir. Örneğin;

S ★★ 2.5		
S ★★ 3.0	ya da	S ✶✶ 3
S ★★ (5./2.)	ya da	S ✶✶ (5/2)

gibi. Ancak, S ** (5./2.) ve S ** (5/2) işlemlerinin sonuçlarının farklı olacağı anımsanmalıdır. Çünkü 5./2. = 2.5 iken 5/2=2 dir.

Fakat negatif gerçel bir sabit ile tamsayılı bir sabit gerçel bir üsse yükseltilemez. Örneğin;

−S ** 2.5 ve 5 ** B

gibi ifadeler geçerli değildir.

6. A ** B ** C gibi bir işlem ancak IBM System/360'da geçerlidir ve sağdan sola doğru değerlendirilir. Yani, $a^{(b^c)}$ ifadesi gibi işleme konulur. Eğer $(a^b)^c$ cebirsel ifadesi gibi bir işlem düşünülüyorsa FORTRAN karşılığı (A ** B) ** C biçiminde yazılmalıdır. Çünkü (A**B) **C işleminin sonucu A**(B**C) işleminin sonucundan farklıdır.

7. Gerçel bir aritmetik işlem sonucu bir tam sayı değişkenine verilirse sonuç tam sayı olarak alınır. Yani kesir kısmı atılır. Örneğin;

I=7.0/2.0 işleminde sonuç 3.5 olarak değil,

I=3 bulunur.

2.5 Aritmetik Atama Deyimleri

Bir aritmetik atama deyimi, sol tarafta bir değişken ondan sonra bir eşittir (=) işareti ve bu işaretin sağında bir aritmetik ifadeden oluşur. Örneğin;

HASLAT=SATIS * FIYAT

ifadesi bir aritmetik atama deyimi olup SATIS değişkeninin değeri ile FIYAT değişkeni değerinin çarpılıp sonucun da HASLAT değişkeni adı altında bellekte depolanmasını belirtir. Başka bir anlatımla yukarıda verilen atama deyimi HASLAT eşittir SATIS çarpı FIYAT anlamını doğrudan vermez. Bu nedenle;

A=A * 5.0 ya da A=A+B

gibi bir aritmetik ifade (atama deyimi) cebirsel olarak anlamsız olduğu halde FORTRAN'da geçerli ve sık sık kullanılan bir ifade biçimidir. Çünkü, örneğin A=A+B atama deyiminin belirttiği : A değişkeni değeri ile B değişkeni değerinin toplanıp sonucunda yine A değişkeni adı altında depolandığıdır. Böylece A değişkeninin değeri değiştirilmekte ve yeni bir değer almaktadır. Yani A değişkenine yeni bir de-

ğer atanmaktadır. O halde A=A/2.0 ya da A=B gibi bir ifade de aynı düşünce ile geçerli bir atama deyimidir.

Yukarıda da açıklandığı gibi eşit işaretinin sol tarafında yalnızca bir değişken bulunmalıdır. Bu nedenle;

SATIS ★ FIYAT = HASLAT

gibi bir ifade geçerli değildir. Ayrıca diğer bir kural olarak eşitliğin sağ tarafında yer alacak olan aritmetik ifadedeki değişkenler işleme konulmadan önce sayısal değerinin bilgisayara bildirilmesi gerekir. Kuşkusuz, aritmetik ifadedeki değişkenlerin sayısal değeri önceden bilinmezse işlem yapılamaz. Örneğin ;

FIYAT=5.0

SATIS=20.0

HASLAT=SATIS ★ FIYAT

ifadeleri ile HASLAT = 20.0 ★ 5.0 = 100.0 olarak bellekte depolanır.

2.6 Girdi/Çıktı Deyimleri

Girdi ve çıktı deyimleri, oku komutunu veren READ ve yaz komutunu veren WRITE deyimleridir. READ deyimi ile bilgisayarın verileri okuması WRITE deyimi ile de işlem sonuçlarını yazıp vermesi sağlanır.

2.6.1. READ ve WRITE Deyimleri

READ deyiminin genel kalıbı aşağıdaki biçimdedir.

READ (i,n) < değişken listesi >

Burada;

i : Verilerin bilgisayara hangi girdi biriminden bilgisayara aktarılacağını belirtir ve bilgi işlem merkezince saptanan bir tamsayıyı temsil etmektedir. READ deyimi ile bundan sonraki bölümlerde yapılacak örneklerde kart okuyucusuna karşılık olmak üzere i=5 varsayılacaktır.

n : Tamsayılı bir sabit olup verilerin kartlardan nasıl okunacağını belirten FORMAT deyiminin numarasıdır. Bu deyim biraz sonra açıklanacaktır.

< değişken listesi > : Birbirinden virgül ile ayrılmış değişkenleri belirtir. Böylece;

READ (5,40) KYIL, FAIZ, SERMAY

biçiminde yazılan bir READ deyimi : 40 numaralı FORMAT deyimine göre KYIL, FAIZ ve SERMAY değişkenlerinin sayısal değerlerinin 5 numaralı kart okuyucusu tarafından okunacağını belirtir.

Yazdırma işlemini yapan WRITE deyiminin de genel kalıbı benzer biçimde aşağıdaki gibidir.

WRITE (i,n) < değişken listesi >

Burada da;

i : Bir tamsayı olup program sonucunun hangi çıktı biriminden verileceğini temsil etmektedir. Bilgisayar merkezlerince saptanan bu tamsayı örneklerimizde, program sonuçlarını standart bilgisayar kağıtları üzerine satır satır yazan çıktı birimine (printer) karşılık olmak üzere 6 varsayılacaktır. Hemen belirtilmelidir ki okutma ve yazdırma birimlerini belirleyen i sayıları onları kullanan herkes tarafından aynı sayı olarak alınır.

< değişken listesi > : Birbirlerinden virgül ile ayrılmış ve değerleri yazdırılmak istenen değişkenleri belirtir.

n : Program sonuçlarının (çıktının) kağıt üzerine nasıl yazılacağını belirten FORMAT deyiminin numarasını temsil eden bir tamsayı.

Örneğin;

WRITE (6,50) TUTAR

ifadesi : 50 numaralı FORMAT deyimine göre 6 numaralı yazıcı aracılığı ile TUTAR değişkeni değerinin yazılacağını göstermektedir.

2.6.2 FORMAT Deyimi

READ ve WRITE deyimlerini tamamlayan FORMAT deyiminin genel kalıbı aşağıdaki biçimdedir.

n FORMAT (s_1,s_2,s_3, ..., s_m)

Burada;

n : READ ve WRITE deyimlerinde belirtilmiş olan ve FORMAT deyiminin numarasını gösteren bir tamsayı sabittir.

s_j : Bu dizi okuma ya da yazma işleminin nasıl yapılacağını belirten FORMAT belirleyicilerini temsil etmektedir. j=1,2,...,m.

Biraz önce verilen örnekler gözönüne alınırsa tamamlanmış READ ve WRITE deyimleri aşağıdaki gibi yazılabilirler.

```
      READ (5,40) KYIL, FAIZ, SERMAY
40    FORMAT (s1,s2, ....., sm)
      .............
      .............
      WRITE (6,50) TUTAR
50    FORMAT (s1,s2, ....., sm)
```

Girdi ve çıktı deyimlerini tamamlayan FORMAT deyimi READ ve WRITE deyimlerinden hemen sonra yazılabileceği gibi programın herhangi bir yerine de konulabilir. s_j simgesi ile gösterilen FORMAT belirleyicileri aşağıda ayrı bir alt bölüm altında önce READ sonra da WRITE deyimi için ayrı ayrı açıklanacaklardır.

2.6.2.1. FORMAT Belirleyicileri

Önemli FORMAT belirleyicileri aşağıdaki gibi ayrı ayrı incelenebilir.

1. **Iw Belirleyicisi** : Bu belirleyici w kadar basamaklı ve tamsayı değerli bir tamsayı değişkeninin okunacağını ya da yazılacağını belirtir. Örneğin;

```
      READ (5,40) KYIL
40    FORMAT (I2)
```

deyimleri ile KYIL değişkeni için veri kartından iki sütun okunur. Veri kartlarının 80 sütun olduğu ve tüm sütunların veri yazmak (delmek) için kullanılabileceği anımsanmalıdır. Yukarıdaki ifadeye göre bilgisayar veri kartının 1. ve 2. sütununa yazılmış tamsayı değeri ne ise okur ve KYIL değişkeni için depo eder.

Yazma durumunda ise w kadar karakterlik yer ayrılır ve değişkenin depo edilen değeri yazılır.

İkinci bir örnek olarak : N=14, J=241, K=23 ve L=52 olduğunu varsayalım ve verilerin veri kartının sütunlarına;

1	2	3	4	5	6	7	8	9	..... 80 (sütun)
1	4	2	4	1	2	3	5	2	

biçiminde delindiğini varsayalım. Bu değişkenlerin değerini okutmak için de;

```
      READ (5,28) N,J,K,L
28    FORMAT (I2, I3, 2I2)
```

ifadelerinin yazılmış olduğunu düşünelim. Buna göre bilgisayar 28 nolu FORMAT deyimine göre veri kartının ilk iki sütunundan N'nin değerini (14); 3.,4. ve 5. sütunlardan J'nin (241); 6. ve 7. sütunlardan K'nın (23); ve son olarakda 8. ve 9. sütunlardan L'nin değerini (52) okuyacak ve depo edecektir. Görüldüğü gibi K ve L değişkenleri için aynı belirleyici 2 kez tekrarlanmakta, birinci sefer K'nın ikinci sefer L'nin değeri okunmaktadır. Belirleyicinin tekrarlama sayısı önüne yazılmaktadır, 2I2 gibi. Kuşkusuz bu durum ancak birbirini izleyen ve aynı sayıda basamak değerine sahip değişkenler için olasıdır. Ayrıca belirtilmelidir ki 2I2 belirleyicisi I2, I2 biçiminde ayrı olarak da yazılabilir.

Okutma işlemi yapılırken veri kartında boş bırakılan sütunlar sıfır (0) olarak okunur. Örneğin, K=34 ve L=1260 ise ve bu verilerde karta;

1	2	3	4	5	6	7	..... 80 (sütun)
3	4	b	1	2	6	b	

biçiminde delinmiş ise (b sütunun boş bırakıldığını göstermektedir);

```
      READ (5,20) K,L
20    FORMAT (I3, I4)
```

deyimi ile K değişkeninin değeri 340 ve L'nin değeride 1260 olarak okunur. K için I3 ile ilk üç sütun okunuyor. Ancak üçüncü sütun boş (b) bırakıldığından sıfır olarak alınacak ve sonuç olarak gerçek değeri 34 olan K değişkeni 340 değerini alacaktır. Böylece hatalı bir okutma işlemi yapılmış olmaktadır. L değişkeni için değişen bir şey yoktur. 4 sütun okunuyor. Yedinci sütun boş olduğu için sıfır olarak alınıyor ve böylece de kartta 126 yazılmış olmasına karşın L değişkeni 1260 değerini almaktadır. Bu nedenle veri okutma işleminde boş bırakılan sütunların okutulan değerin sağında bırakılması durumunda dikkatli olmak gerekir .Örneğin K değeri karta b34 biçiminde delinebilir.

2. **Fw.d Belirleyicisi :** Bu belirleyicide ondalıklı biçimde olan gerçel sayıların okutulması ya da yazdırılması için kullanılır. w harfi sayısal değerin kaç basamaklı olduğunu d harfide gerçel sayının ondalık noktasından sonra kaç basamak bulunduğunu belirtir.

Böylece, eğer Fw.d belirleyicisi READ deyiminde kullanılmışsa w kadar sütun okunacak ve bu sütunlarda okunan sayının d kadar basamağı da ondalık noktasından sonra yer alacaktır. Örneğin;

```
1  2  3  4  5  6  7  8  9 ..... 80 (sütun)
4  8  2  6  b  3  2  1  b
```

biçiminde delinmiş bir veri kartından,

```
      READ (5,35) A,B
35    FORMAT (F5.3, F3.2)
```

deyimleri ile A değişkeni için 5 sütun yani 48260 sayısı okunacak ve bu sayının da sonundan üç basamağı (d=3) ondalık noktasından sonra yer alacaktır. Böylece A değişkeni 48.260 olarak bilgisayar belleğinde depo edilecektir. Aynı şekilde B değişkeni de 3.21 olarak depo edilecektir.

Yukarıdaki örnekte görüldüğü gibi ondalık gerçel sayıları noktalı olarak veri kartına delme zorunluluğu yoktur. Ondalık noktanın yerini d belirlemektedir. Eğer veriler noktalı olarak veri kartına delinmiş ise ve noktadan sonraki basamak sayısı ile d'nin belirttiği basamak sayısı farklı ise, veri kartına delinen noktanın yeri geçerlidir. Örneğin veriler karta

```
1  2  3  4  5  6  7  8  9 ..... 80 sütun
8  4  2  5  1  4  3     8
```

biçiminde delinmiş ise;

```
      READ (5,25) A,B
25    FORMAT (F4.3, F5.2)
```

deyimleri ile A değişkeni için 8.425 ve B değişkeni için de 143.8 değeri okunur. Görüldüğü gibi belirleyici F5.2, B için noktadan sonra iki basamak belirttiği halde karta delinen noktanın yeri geçerli olmakta d'nin belirttiği gözönüne alınmamaktadır.

Başka bir örnek olarak A=37.556, B=486.4 ve C=25. olduğunu ve veri kartına aşağıdaki biçimde delindiğini varsayalım.

```
123456789 ..... 80 (sütun)
bb37556b486.4b25
```

16 sütuna yazılmış bu verileri okutmak için

```
      READ (5,35) A,B,C
35    FORMAT (2F7.3,F2.0)
```

deyimleri yazılmış olsun.

Buna göre 2F7.3 belirleyicisi ile A ve B değişkenleri için ayrı ayrı 7'şer sütun okunacaktır. Yani F7.3 iki kez tekrarlanacaktır. Bu durumda A için okunan ilk 7 sütunda son üç basamak noktadan sonra yer alacak ve A değişkeni 37.556 (0037.556) olarak depolanacaktır. İkinci kez 7 sütun okunduğu zaman B'nin değeri 486.40 olarak depolanacaktır. Görüldüğü gibi karttaki ondalık noktanın yeri geçerli olmaktadır. Sonkez F2.0 belirleyicisi ile iki sütun okunacak ve noktadan sonra basamak bulunmayacaktır. Böylece C değişkeni 25. olarak okunacaktır.

3. **Ew.d Belirleyicisi** : E üslü biçiminde yazılmış olan gerçel sayıları okutmak ya da yazdırmak için kullanılır. w harfi sayının kaç basamaklı (mantis, işaret ve üs dahil) ya da karakterli olduğunu, d harfide ondalıklı noktanın mantisin (E'den önceki kısmın) neresinde olduğunu yani mantiste noktadan sonra kaç basamak buluduğunu belirtir. Fw.d belirleyicisinde olduğu gibi eğer karta delinmiş sayının noktası ile Ew.d belirleyicisinde d'nin belirttiği noktanın yeri farklı ise karta delinmiş noktanın yeri geçerlidir. Mantis noktasız yazılmış ise noktanın yerini d belirler. Örneğin, veriler kartta;

123456789 ..... 80 (sütun)
426E314.5E—b2

biçiminde delinmiş ise ve

```
      READ (5,10) A,B
10    FORMAT (E5.2,E8.2)
```

deyimleri ile okutuluyorsa A=4.26E3=4260. ve B=14.5E—02=0.145 değerleri ile depo edileceklerdir. Görüldüğü gibi B değişkeni için noktanın yeri kartta yazılan (delinen) yerdir. Gerçel sayılar hangi biçimde yazılırsa yazılsın ondalık olarak depo edilirler.

E için geçerli kurallar çift hassaslıklı sayılarda D harfi kullanarak aynıdır. Yani belirleyici Dw.d biçimindedir.

4. **nX Belirleyicisi** : Bu belirleyici READ deyimi için FORMAT'ta kullanılmış ise okuma işlemi karttan yapılırken bilgisayar n kadar sütunu atlar, yani okumaz. Eğer WRITE deyimi için kullanılmış ise çıktının bilgisayar kağıdına yazılması sırasında bilgisayar (n) kadar karakterlik yeri atlar, yani boş bırakır.

Aşağıda verilen veri kartından A,I ve C değişkenleri için okutma işlemi yapıldığını varsayalım.

123456789 ... ... 80 (sütun)
bb4825b786bbb34bbb8464.

```
      READ (5,15) A,I,C
15    FORMAT (3X,F6.2,4X,I2,3X,F5.2)
```

Buna göre, bilgisayar ilk önce 3 sütun atlar ve F6.2 gereği A için 8250.78 okunur. Sonra 4 sütun atlar ve I2 gereği I değişkeni için 34 değeri okunur. Tekrar 3 sütun atlar ve F5.2 gereği C için 5 sütun okur. C'nin verileri karta noktalı olarak delindiği için karttaki noktanın yeri esas alınır ve C=8464. olarak okunur ve depolanır.

5. " / " **Belirleyicisi**. Bölüm ya da kesme işareti biçiminde gösterilen bu belirleyici READ deyimi için kullanıldığında, bir veri kartı ya da okuma işlemi yapılıyorsa okunmakta olan kartın geri kalan kısmı atlanır. Eğer WRITE deyimi için belirtilmişse bilgisayar yazma işleminde bir satırı ya da yazma işleminin yapıldığı satırın geri kalan kısmını atlar ve bir sonraki satıra geçer.

Örneğin aşağıdaki READ deyimi ile ayrı ayrı kartlarda delinen verilerin okutulmak istendiğini düşünelim.

```
      READ (5,50) A,B,I,HAS
50    FORMAT (/,F4.0,3X,F3.0/,I2,/ /,8X,E5.0)
```

veriler :

```
      123456789 ... ... ... ... ... ... ... ... ... ... ... 80 sütun
1. kart :  b43.2bb847321869
2. kart :  2b8360b94254303.
3. kart :  846320b483
4. kart :  6b42863b342E3
5. kart :  742138b9430E2567
```

Okuma işlemi başladığı zaman, 1. veri kartı atlanacak ve A değişkeninin değeri 2. kartın ilk dört sütunundan 2083. olarak okunacaktır. Sonra üç sütun atlanacak ve B değişkeninin değeri F3.0 belirleyicisine göre 942. olarak okunacak ve bu kartın geri kalan sütunları atlanıp 3. kartın okunmasına geçilecektir. Bu karttan I2 gereği ilk iki sütun I değişkeni için okunacak ve I=84 olarak depolanacaktır. Sonra / / belirleyicilerinin gereği olarak 3. kartın geri kalan sütunları ile 4. kart atlanacak ve bilgisayar okuma işlemi için 5. karta geçecektir. Bu kartında ilk sekiz sütunu 8X gereği atlanacak ve HAS değişkeni için E5.0 belirleyicisine göre beş sütun okunacak ve HAS değişkeni değeri 430.E2=43000. olarak bellekte depo edilecektir. Sonuç olarak :

A=2083.

B=942.

I =84

HAS=43000. değerlerini alacaklardır.

6. **Aw Belirleyicisi :** Sayısal değerlerden (nümerik karakterlerden) ayrı olarak kimi zaman bilgisayarın belleğinde, alfabetik harfler ve diğer özel semboller gibi karakterlerin depo edilmesi ve gerektiğinde program çıktısı olarak yazdırılması gerekebilir. Bu tür "alfa-nümerik" adı verilen karakterleri okutmak ve yazdırmak için Aw belirleyicisi kullanılır. w harfi yine karakterlerin sayısını temsil eden bir tamsayıdır.

Alfa-nümerik karakter dizilerini okutmak için sayılarda olduğu gibi değişken isimleri oluşturulur. Bu değişkenler için gerçel ya da tamsayılı ayırımına gerek yoktur. Çünkü bu değişkenlerin programda kullanılmalarındaki özel durum ve Aw belirleyicisi onları diğer değişkenlerden ayırır.

Aw ile belirtilen alfa-nümerik değişkenlerin alacakları karakter sayısı bilgisayarın alfa-nümerik karakterleri depo edeceği yere bağlı olup 4 ile 8 karakter arasında değişir. Bu sayıyı v harfi ile gösterirsek, IBM System/360 için v=4 karakterdir. Yani karakter dizilerinin okunması yani depolanması için her alfa-nümerik değişkene 4 karakterlik yer ayrılır.

Okutma işleminde eğer w < v ise w kadar olan karakterler ile geriye kalan (v—w) kadar karakterlik yer boş olarak depolanır. Ancak w > v ise w kadar olan karakterlerin en sağdaki v kadarı depo edilir geriye kalan soldaki (w—v) kadar karakter ise gözönüne alınmaz.

Örneğin, bir işletme aylık ücret bordrosunu hazırlarken işçilerin adı, soyadı, sicil numaraları ve günlük ücretlerini belirten bilgileri bilgisayara okutmak istiyor. Bu bilgiler kartlara aşağıdaki biçimde deliniyor :

1— 5. sütunlara	Adlar,
6. sütun	Boş (b),
7—11. sütunlara	Soyadlar,
12—13. sütunlar	Boş (b),
14—16. sütunlara	Sicil numarası,
17—20. sütunlar	Boş (b),
21—25. sütunlara da	Günlük ücretler.

Bu çerçeveye göre delinmiş bir veri kartı,

123456789 ... ... 80 (sütun

HASANbGEZENbb37Bbbbb42460

biçiminde olsun. Bu bilgileri okutmak için de,

```
    READ (5,20) AD, SOYAD, SICNU, UCRET
20  FORMAT (A5, 1X, A5, 2X, A3, 4X, F5.2)
```

ifadelerinin yazıldığını varsayalım. Buna göre okuma işlemi şöyle olacaktır : ,

AD değişkeni için A5 gereği beş sütun okunacak. Ancak en sağdaki dört karakter (v=4) bellekte depo edilecek ve soldaki bir karakter (w—v=5—4=1), yani H harfi depo edilmeyecektir. Böylece AD değişkeni için ASAN karakter dizisi depo edilir. Sonra 1X gereği bir sütun atlanır ve SOYAD değişkeni için de A5 belirleyicisine göre okuma işlemi yapılır. Aynı şekilde SOYAD değişkeni için EZEN karakter dizisi depo edilir. Yani G karakteri depolanmaz.

Daha sonra 2X gereği iki sütun atlanır ve SICNU değişkeni için A3 gereği üç sütun okunur. Depolanırken dört karakterlik depolama bölgesinde (v=4) son bir karakterlik yer (v—w=4—3=1) boş olarak depo ediliyor. Yani SICNU değişkeni değeri 37Bb karakter dizisi olarak depo ediliyor. Son olarak dört sütun daha atlanır ve UCRET değişkeni için F5.2 gereği 424.60 okunur. Böylece;

AD değişkeni : ASAN
SOYAD » EZEN
SICNU » 37Bb
UCRET » 424.60 olarak depolanır.

Alfa-nümerik değişkenlerin okutulması işleminde gözönünde bulundurulması gereken kurallar aşağıdaki biçimde özetlenebilir.

Veriler	:	ABC	ABCD	ABCDEFG
Belirleyici	:	A3	A4	A6
Depolama	:	|A|B|C| |	|A|B|C|D|	|D|E|F|G|

Alfa-nümerik değişkenleri yazdırma işleminde eğer w < v ise (v depolanmış karakter sayısı), yani yazma işlemi için ayrılan yer sayısı depolanmış karakter sayısından az ise, soldaki w kadar karakter yazılır geriye kalan sağdaki v—w kadar karakter yazılmaz. Eğer w > v ise v kadar depolanmış karakter en sağda yazılır geriye kalan w—v kadar karakter yeri satırın sol tarafında boş bırakılır.

Örneğin biraz önce READ deyimi ile AD ve SOYAD alfa-nümerik değişkenleri için depolanan karakter dizilerini (ki bunlar sırası ile ASAN ve EZEN'di yazdırmak için;

```
      WRITE (6,15) AD, SOYAD
15    FORMAT (3X, A3, 2X, A5)
```

deyimlerinin yazıldığını varsayalım. Buna göre çıktı işlemi şöyle olacaktır.

İlk önce çıktı kağıdında üç karakterli yer atlanacak (boş bırakılacak) sonra ASAN olarak bellekte depolanan AD değişkeninin karakterleri A3 gereği üç karakterlik yere ASA olarak yazılacaktır. Çünkü $w < v$ dir. Böylece ASAN'nın son karakteri olan N yazılmayacaktır. Daha sonra 2X gereği iki karakterlik yer yine boş bırakılacak ve SOYAD değişkeni için A5 belirleyicisine göre kağıtta ayrılan beş karakterlik yerin ilk karakter yeri ($w-v=5-4=1$) boş bırakıldıktan sonra v kadar karakter sağda yazılacaktır. Yani SOYAD değişkeni bEZEN olarak yazılacaktır. Böylece 132 karakter yerlik çıktı kağıdında yukarıdaki yazdırma işleminin çıktısı şöyle olacaktır.

```
123456789 ... ... ... ... ... ... ... ... ... ... 132  (yer)
bbbASAbbbEZEN
```

Çıktı işlemi ile ilgili bu kurallar şöyle özetlenebilir :

Depolama (bellekte)	:	ABCD	ABCD	ABCD	AB
Belirleyici	:	A3	A4	A6	A4
Çıktı	:	|A|B|C|	|A|B|C|D|	| | |A|B|C|D|	|A|B| | |

Alfa nümerik değerleri okutma ve yazdırma işlemlerinde doğru bir yaklaşım tüm karakterleri birlikte düşünmek ve yer sayısına göre yeteri kadar Aw belirleyicisi kullanmaktır (sayfa 60'a bakınız).

7. **wH Belirleyicisi** : Bilgisayar programının sonucu (çıktı) genellikle 132 karakterlik yeri olan ve 60 satırdan oluşan bilgisayar kağıtlarına yazılır. Her satırın birinci karakter yeri yazılmaz ve kağıdın ileriye doğru haraketini sağlamak için bir denetim aracı olarak kullanılır. Yani yazma işleminin normal satır aralığında mı ya da ayrı ayrı sayfalarda veya çift satır aralığında mı yapılması gerektiğini denetler.

Denetleme için kullanılan karakterler ve anlamları aşağıda verilmiştir :

b (boş) : Yazma işleminden önce kağıt normal satır aralığı kadar ileriye gider.

0 (sıfır) : Yazma işleminden önce kağıt çift satır aralığı kadar ileriye gider.

1 : Yazma işleminden önce yeni bir sayfaya geçilir.

+ : Kağıt hareket etmez. Yani aynı satırda kalır.

Yazma işleminde bu denetimi sağlamak için H belirleyicisi kullanılır ve FORMAT'ta (1Hb), (1H0), (1H1) ve (1H+) biçiminde yazılarak belirtilir. Örneğin;

```
      WRITE (6,10) A,B
10    FORMAT (1H0, F8.3, 2X, F5.0)
```

deyimi ile A ve B değişkenlerinin sayısal değerleri yazılmadan önce kağıt iki satır ileriye gider. 1H0 belirleyicisi ile ilk karakter 0 (sıfır) yapılmıştır ve bu da bilgisayara iki satır ilerlemesini bildirir. Aynı şekilde;

```
      WRITE (6,15) A,B
15    FORMAT (1H1, F8.3, 2X, F5.0)
```

deyimi de A ve B'nin sayısal değerlerini yeni bir sayfanın başına yazar. Böylece ilk karakter 1 yapılmakta ve bu da yazda işleminin yeni bir sayfada yapılmasını sağlamaktadır.

Normal satır aralığında yazma işlemi için ilk karakterin boş (b) yapılması 1Hb biçimi dışında bir kaç biçimde yapılabilir. Örneğin; nX belirleyicisini ilk FORMAT belirleyicisi olarak kullanmakla birinci karakter yer boş yapılabilir. Böylelikle yazma işleminin normal satır aralığında yapılması sağlanır.

Örnek :

```
      WRITE (6,20) I,N
20    FORMAT (3X, I3, 2X,I5)
```

Bu deyimde ilk belirleyici 3X ile birinci karakter b (boş) yapılmakta ve yazma işleminin normal satır aralığında yapılması sağlanmıştır. Böylece kağıt normal satır aralığı kadar ilerledikten sonra üç karakter boş (b) bırakılacak ve ondan sonrada diğer belirleyicilerin gereği ya-

pılacaktır. Yani çıktıda karakter yerlerinin dağılımı aşağıdaki gibi olacaktır.

123456789 ... ... 132 (yer)
bbbxxxbbxxxxx

Diğer yandan ilk karakteri b yapmak için ikinci bir yöntem de değeri yazılacak ilk değişken için fazla karakter yeri vermektir. Örneğin, yukarıdaki örnekte olduğu gibi eğer I değişkeninin sayısal değerinin üç basamaktan fazla olmayacağı düşünülüyorsa I değişkenine üçten fazla yer ayırmak suretiyle ilk karakterin bir boşluk olduğu, yani normal satır aralığında yazması gerektiği bildirilir. Böylece;

```
      READ (6,10) I,N
10    FORMAT (I5, 2X, I5)
```

deyimi ile yazma işleminden önce kağıt normal satır aralığında ilerleyecektir. Çünkü ilk beş karakterlik yer I için ayrılmıştır. Fakat I değişkeninin sayısal değeri üç basamaktan fazla olmayacağı varsayıldığından ilk iki karakterlik yer boş kalmaktadır. Böylece ilk karakterin boş olması kağıdın normal satır aralığında ilerlemesini sağlamaktadır. Çıktıda karakter yerlerinin dağılımı aşağıdaki gibi olmaktadır.

123456789 ... ... 132 (yer)
bbxxxbbxxxxx

Görüldüğü gibi H belirleyicisine gerek kalmadan ilk karakter boş yapılmakta ve böylece kağıdın normal satır aralığında ilerlemesi sağlanmaktadır.

H belirleyicisi ayrıca harf, sayı ve özel karakterlerden (boş yer dahil) oluşan karakter dizilerini çıktıda yazdırmak için de sık sık kullanılır. Bu belirleyicinin genel kalıbı wH olarak belirtilir. w harfi H'dan sonra yazdırılması istenen karakter dizisinin karakter sayısını belirtmektedir. Örneğin GSMH 1981 ifadesini yazdırmak istersek;

```
      WRITE (6, 15)
15    FORMAT (10HbGSMHb1981)
```

deyimi yazılabilir. H'dan önceki 10, on karakterin olduğu gibi yazılacağını belirtmektedir. İlk boş (b) karakter yazma işleminin normal satır aralığında yazılacağını ifade etmektedir. GSMH'dan sonraki boşluk (b) ise GSMH'dan sonra arada bırakılacak boşluğu belirtmektedir. Tüm boşluklar karakter saymada hesaba katılacaktır. Böylece çıktı :

```
123456789 ... ... ... ... ... ... ... ... ... 132 (yer)
 GSMH 1981
```

olarak yazılmaktadır. Aynı çıktı

```
15 FORMAT (1X, 9HGSMHb1981)
```

deyimi ile de yazılabilir.

Yazdırılmak istenen karakter sayılarını teker teker sayılmasını ve ondan sonra bu sayının (w) H harfi önüne yazılmasını önlemek için yani bu sayma işleminden kurtulmak için karşılıklı iki (' ') işaret de kullanılmaktadır. Bu daha çok kolaylık sağlamaktadır. Yukarıdaki yazma işlemi aşağıdaki biçimde daha kolay olarak yapılır.

```
    WRITE (6,15)
15  FORMAT ('bGSMHb1981')
```

ya da

```
15  FORMAT (1X,'GSMHb1981')
```

Her iki durumda da çıktı yukarda verilenin aynısıdır. Aynı şekilde (1H0), (1Hb), (1H1) ve (1H+) denetim karakterleride ('0'), ('b'), ('1') ve ('+') biçiminde yazılabilir. Örneğin;

```
    WRITE (6,15)
15  FORMAT ('b', 'GSMHb1981')
```

deyimi ile aynı çıktı yazdırılacağı gibi,

```
    WRITE (6,15)
15  FORMAT ('1', 'GSMHb1981')
```

deyimi ile de aynı çıktı yeni bir sayfa başında yazılmaktadır.

Yukarıdaki örneklerde WRITE deyimi ile değeri yazdırılmak istenen bir değişken listesi yoktur. Bu tür deyimler yalnızca karakter dizisi yazdırmak için yazılmışlardır. Değeri yazdırılmak istenen değişken olsaydı listeye alınır ve belirleyicileri de FORMAT'ta belirtilerek yazdırılabilir. Bu konu aşağıda açıklanacaktır.

Çıktıda FORMAT Belirleyicileri : Daha önceki sayfalarda belirtildiği gibi girdi ve çıktı için FORMAT belirleyicileri aynıdır. Ancak tamsayılı ve gerçel değişkenlerin sayısal değerlerini yazdırmak için yer sayısının (w) belirlenmesinde bazı noktaların göz önünde bulundurulması gerekir.

1. Iw belirleyicisi ile yapılacak yazdırmalarda sayısal değerin işareti için bir karakterlik yer ayrılmalıdır. Örneğin bellekte 3495 de-

ğeri ile depo edilmiş bir K değişkeninin değeri yazdırılmak istenirse enaz bir I5 belirleyicisinin FORMAT'ta yer alması çok uygun olacaktır. Böylece hesaplama sonucu bulunan ve değeri yazdırılacak olan sayıların negatif olası durumunda (—) işareti için de bir yer ayrılmış olur.

2. Fw.d belirleyicisi ile yapılacak yazdırmalarda sayının işareti ve ondalık noktası içinde birer yer ayrılmalıdır. Örneğin, 1425.6264 olarak depolanmış bir sayıyı yazdırmak için en az w=10 olmalıdır, F10.4 gibi.

1'den küçük sayılar için ondalık noktasından önce bir 0(sıfır) yazılır, noktadan sonra (d) kadar basamak yazılır. Eğer depolanmış olan sayının kesir kısmının basamak sayısı (d)'den fazla ise (d) kadar basamak yuvarlama yapılarak yazılır, geriye kalan sağdaki kesirli basamaklar yazılmaz. Örneğin 1425.6264 sayısını F10.2 ile yazdırmak istersek bu sayı bbb1425.63 olarak yazılır. Eğer sayının kesirli basamak sayısı (d)'den az ise (d) kadar kesirli basamak sayısını tamamlamak için kesir kısmından sonra gerekli sayıda 0 yazılır. Örneğin 1425.6264 olarak depolanan sayıyı F12.6 ile yazdırırsak bu sayı b1425.626400 olarak yazılır.

3. Ew.d ve Dw.d belirleyicileri ile yazdırma yapılırken aşağıdaki sıra izlenir :

a) İlk önce negatif işareti yazılır. Eğer işaret pozitif ise bir boşluk bırakılır.
b) Bir 0 (sıfır) yazılır.
c) Sıfırdan sonra ondalık nokta konulur.
d) Kesirli kısmın (d) kadar basamağı yazılır.
e) E harfi (ya da çift hassaslıklı istenmiş ise D) yazılır.
f) Üs işareti yazılır. Ancak işaret pozitif ise boşluk bırakılır.
g) Son olarak iki karakterlik yere üs yazılır.

Böylece yukarıdaki her sıra için gerekli olan karakter yerleri toplanırsa (d) kadar kesirli kısımdan ayrı olarak 7 karakterlik yer gerekmektedir. Buna göre Ew.d ve Dw.d belirleyicilerinde genel olarak w=7+d olmalıdır. Örneğin 4 kesir basamaklı bir sayıyı yazdırmak istediğimizde w=7+4=11, yani E11.4 olmalıdır. Çift hassaslıklı durumlarda örneğin tüm anlamlı 16 basamak isteniliyorsa w=7+16=23, yani D23.16 belirleyicisi yazılacaktır.

4. Iw, Fw.d, Ew.d ve Dw.d belirleyicileri ile yazdırmada depolanmış sayının basamak sayısı ayrılan yer sayısından (w) az ise depolan-

mış sayı sağda kalacak biçimde yazılır, ayrılan yerin artakalan kısmı boş olarak solda bırakılır. Örneğin, 415.58 olarak depolanmış bir sayı F8.2 belirleyicisi ile yazdırılırsa çıktı : bb415.58 olarak yazılır.

5. Depolanmış sayının basamak sayısı ayrılan yer sayısından (w) fazla ise sayısal değer yazılmaz ve ayrılan yerlere ★ (asteriks) işaretleri konulur. Böylece yazdırma için yeterli sayıda yer ayrılmadığı belirtilir. Örneğin, 415.58 sayısını F5.2 ile yazdırmak istesek bu yerlere beş tane ★★★★★ işareti yazılır. Bu durumlarla karşılaşmamak için (w) yeteri kadar geniş belirlenmelidir.

Yazdırma ile İlgili Örnekler :

I = 42	A = 345.53
K = 34	B = 642.0
J = 7	C = 24.386

olarak bilgisayar belleğinde depolanmış olduğunu varsayalım ve yazdırma için aşağıdaki WRITE deyimlerinin yazıldığını göz önüne alalım :

```
1)        WRITE (6,10) A,K
     10   FORMAT (3X, F8.3, 2X, 'K=', I3)

2)        WRITE (6,15) I,C
     15   FORMAT ('bALTbALTAbYAZMA',/1X, 'I=', I3,/,
         ★1X, 'C=', F8.2)

3)        WRITE (6,100) B,J
    100   FORMAT ('bYANbYANAbYAZMA',/, 1X, 'B=', E11.3,
         ★3X, 'J=', I5)
```

Çıktılar :

1 nolu ifadeye göre;

bbbb345.530bbK=b34

2 nolu ifadeye göre;

Birinci satır bALTbALTAbYAZMA
İkinci satır bI=b42
Üçüncü satır bC=bbb24.39

3 nolu ifadeye göre;

Birinci satır bYANbYANAbYAZMA
İkinci satır bB=bb0.642E+03bbbJ=bbbb7

Bir başka örnek olarak okutma ve yazma işlemlerini birlikte inceleyelim. Değerleri aşağıda verilen değişkenleri okuttuktan sonra yazdırmak isteyelim.

L=83
S=459.576
P=165.25E−2

Bu verilerin iki veri kartına aşağıdaki biçimde delindiğini varsayalım.

123456789 ... ... 80 (sütun)
1. kart 83bbb459.576
2. kart bb16525E−2

Okutma işlemi için aşağıdaki ifadeyi yazalım.

```
      READ (5,20) L,S,P
20    FORMAT (I2, F10.0/2X, E8.2)
```

Bu deyim ile I=83, S=459.576 ve P=165.25E−2=1.6525 olarak depo edilecektir. Depo edilmiş bu değerleri yazdırmak için I2 ve E8.2 belirleyicileri yeterli sayıda yer ayırmamaktadır. Gerekli sayıda yer sağlamak için belirleyicilerin (w) yer sayısını büyütelim ve aşağıdaki çıktı deyimini yazalım.

```
      WRITE (6,25) L,S,P
25    FORMAT (I5, F10.3, 3X, E12.5)
```

Bu durum da çıktı şöyle olacaktır :

123456789 ... ... 132 (Yer)
bbb83bbb459.576bbbb0.16525Eb01

Böylece hazırlanan 25 nolu FORMAT ile değişkenleri okutmak istersek verileri veri kartına aşağıdaki biçimde aktarmak gerekir.

123456789 ... ... 80 (sütun)
bbb83bbb459.576bbbbbb165.25E−2

```
      READ (5,25) L,S,P
25    FORMAT (I5, F10.3, 3X, E12.5)
```

deyimi ile değişkenler için aynı değerler okutulup depolanır. Değerleri yazdırmak için de aynı FORMAT yeterli olduğuna göre 25 nolu FORMAT'ı hem okutma hem de yazdırma için kullanabiliriz. Yani;

```
      READ (5,25) L,S,P
      WRITE (6,25) L,S,P
25    FORMAT (15, F10.3, 3X, E12.5)
```

gibi yazabiliriz.

Böylece görüldüğü gibi READ ve WRITE deyimleri için aynı FORMAT kullanılabilir. Ancak okutma ve yazma işlemini istenildiği gibi yapma koşulunu sağlayıp sağlamadığı kontrol edilmelidir.

2.6.2.2 FORMAT Deyimi ile İlgili Ek Bilgiler

Şimdiye kadar READ ve WRITE deyimlerinin incelenmesi aşamalarında belirtilen FORMAT kurallarına ek olarak, girdi/çıktı ifadelerinin hazırlanmasında kolaylık sağlayan aşağıdaki noktalarda göz önünde bulundurulmalıdır.

1. Bir belirleyicinin önüne konulan bir tamsayılı sabit o belirleyicinin kaç kez tekrarlanacağını ya da aynı türden kaç değişkeni okutacağını belirtir. Örneğin;

```
10    FORMAT (5X, 3F4.2, F7.3, 2I3)
```

deyimi Xbelirleyicisinin 5 kez tekrarlanacağını yani beş yer atlanacağını, F4.2 belirleyicisinin üç kez tekrarlanacağını ve I3 belirleyicisininde iki defa tekrarlanacağını belirtmektedir. Başka bir anlatımla yukardaki deyim ile

```
10    FORMAT (5X, F4.2, F4.2, F4.2, F7.3, I3, I3)
```

deyimi aynıdır.

2. Aynı şekilde birbirini izleyen ve tekrarlanması gereken bir grup belirleyici ayraç içine alınabilir ve bir tamsayı sabiti ayracın önüne konularak kaç kez tekrarlanması gerektiği belirtilebilir. Örneğin;

```
15    FORMAT (3X, 2(I2, 2X, F4.2), F8.3)
```

deyimi ile aşağıdaki deyim aynı anlamı ifade eder.

```
15    FORMAT (3X, I2, 2X, F4.2, I2, 2X, F4.2, F8.3)
```

3. Bir READ ya da WRITE deyiminin değişken listesindeki değişken sayısı bu değişkenlere karşılık olan FORMAT deyimindeki belirleyici sayısından az olabilir. Bu durumda fazla olan belirleyiciler gözönüne alınmazlar. Örneğin;

```
      READ (5,20) A,K,B
20    FORMAT (3X, F5.0, I5, 2X, F8.3, F6.2, I4)
```

deyimi ile okutma işlemi yapılırken A'ya karşılık olarak F5.0, K'ya I5 ve B'ye de karşılık olarak F8.3 belirleyicileri alınır. Başka okutulacak değişken bulunmadığı için F6.2 ve I4 belirleyicileri yok varsayılır.

4. Eğer okutulacak ya da yazdırılacak değişken sayısı FORMAT deyimindeki belirleyicilerden fazla ise, değişken ve belirleyicilerin aynı türden olmaları koşulu ile, tüm değişken değerleri okununcaya kadar ya da yazdırılıncaya kadar FORMAT'taki belirleyiciler tekrarlanır. Her defasında bilgisayar FORMAT'ın en sağdaki ayracı ile karşılaşınca yeni bir kart ya da satıra geçerek tekrar FORMAT'ın en sağındaki sol "(", yani en sağdaki açılan ayraçtan okuma ya da yazma işlemine başlar ve tüm değişkenler bitinceye kadar tekrarlar. Örneğin;

```
      READ (5,30) A,B,C,D,E
30    FORMAT (2X, F5.3, F4.2)
```

deyimi ile okutma işlemi şöyle olacaktır :

Önce birinci veri kartından iki sütun atlanır ve A için F5.3, B için de F4.2'ye göre okutma işlemi yapılacaktır. Sonra ikinci veri kartına geçilecek ve bu kartında ilk iki sütunu atlandıktan sonra C için F5.3 ve D için de F4.2'ye göre okuma işlemi yapılacaktır. Daha sonra üçüncü veri kartına geçilecek ve bu veri kartının da ilk iki sütunu atlandıktan sonra E değişkeni için de F5.3 belirleyicisine göre okuma işlemi yapılıp tamamlancaktır. Böylece okuma işlemi üç ayrı kartta yapılmaktadır.

Eğer yukarıdaki deyim aynı FORMAT'lı bir WRITE deyimi olsaydı, aynı biçimde bu kez de üç ayrı satıra yazılacaktır. Yani, A ve B değişkenlerinin değerleri birinci satırda C ve D değişkenlerinin değerleri ikinci satırda ve E değişkenininde değeri üçüncü satırda yazılmış olacaktır.

Yukarıda verilen deyim ile,

```
      READ (5,30) A,B,C,D,E
30    FORMAT (3(2X, F5.3, F4.2,/) )
```

deyimi aynı anlamı ifade eder ve aynı çıktıyı verir.

Başka bir örnek olarak,

```
      READ (5,3) A,B,C
3     FORMAT (2X, F5.3,/, (F4.2) )
```

deyimini alalım. Buna göre A değişkeninin değeri (2X, F5.3) belirleyicisine göre birinci karttan, B değeri F4.2'ye göre ikinci karttan ve C değişkeninin değeride yine F4.2'ye göre üçüncü karttan okunacaktır.

Böylece açıklandığı gibi, bilgisayar en sağdaki kapanan ")" ayraç ile karşılaşınca değişken listesinde okunacak değişken bulunduğu için okunmakta olan kart atlanır ve kontrol FORMAT'ın en sağdaki açılan "(" ayracına döner ve okuma işlemine devam edilir. Kuşkusuz aynı şey satır bakımından yazma işlemi için de geçerlidir.

2.7 Durdurma Deyimleri

Her bilgisayar programında programın sonunu belirten ya da programın bittiğini bilgisayara bildiren bir deyimin bulunması gerekir. Amaç programının işlenmesini durduran deyim STOP deyimidir. Bir programdaki akış yönü seçeneklerine göre birden fazla STOP deyimi bulunabilir.

Kaynak programın bittiğini belirten deyim ise END deyimidir. Bu deyim derleyiciye kaynak programın bittiğini ve başka deyim aramaması gerektiğini bildirir. Bu nedenle programın bittiğini belirten END deyimi programın en son deyimidir. Sonunda END deyimi olmayan bir program eksik bir programdır. Bir programda ancak bir tek END deyimi bulunur.

STOP ve END deyimlerinden ayrı olarak amaç programın işleyişini geçici olarak durduran bir PAUSE deyiminin de bulunmasına karşın, çok hızlı işleyen sistemlerde, örneğin IBM System/360'da, bu deyimin kullanımı zaman kaybı nedeni ile uygun görülmez.

STOP deyiminin genel kalıbı,

```
STOP n
```

biçimindedir. Burada n harfi beş basamaktan fazla olmayan bir tamsayı sabitini temsil etmektedir. Ancak n değerinin belirtilmesi ya da kullanımı isteğe bağlıdır. Bu nedenle de belirtilmez ve yalnızca,

```
STOP
```

olarak yazılır. Fakat bir programda birden fazla STOP deyimi varsa ve programcı program akımının hangi STOP'da durduğunu bilmek isterse STOP deyimlerini numaralayabilir. Yani n'yi belirtebilir.

Programın en son deyimi olan END deyimi ise yalnızca END olarak programın sonunda yazılır.

2.8 Basit Programlama Örnekleri

Örnek 1. Kârageçiş Noktasının Bulunması

Aylık sabit masrafları 40 000 TL. olan bir işletme üreteceği bir birim malı 2500 TL.'ye satmayı kararlaştırmıştır. Bir birim malın değişir birim maliyeti 1800 TL. olduğuna göre, birim değişir maliyetin üretim hacmine bağlı olarak değişmediğini varsayarsak, bu işletmenin kârageçiş noktası nedir? Yani kârı ve zararı sıfır yapan üretim miktarı (hacmi) nedir? Bilindiği gibi bu noktada işletme ne kâr ne de zarar yapmaktadır.

Değişkenlerin Oluşturulması :

SAMAS	:	Sabit masraflar = 40000. TL.
FIYAT	:	Birim satış fiyatı = 2500. TL.
DEMAL	:	Birim değişir maliyet = 1800. TL.
KARGEN	:	Kârageçiş noktası = ?

Görüldüğü gibi oluşturulan değişkenlerin ilk üçü gerçel değişkenler son değişken KARGEN ise bir tamsayı değişkenidir. Buna göre KARGEN değişkeni bir tamsayı sabiti değeri alacaktır.

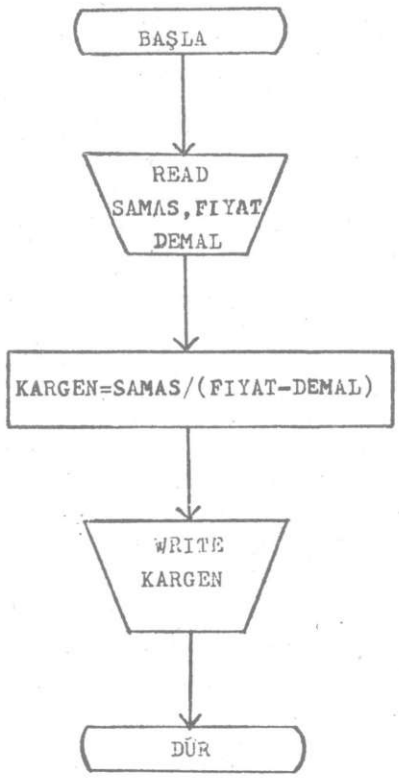
Problemin çözüm yöntemi bilinen formülüne göre,

$$\text{KARGEN} = \frac{\text{SAMAS}}{\text{FIYAT} - \text{DEMAL}}$$

biçiminde ifade edilebilir.

Bu açıklamalar çerçevesinde program akış-şeması aşağıdaki gibi çizilebilir.

Akış - Şeması



Bu akış-şemasına dayalı olarak program şöyle yazılabilir.

```
1 2 3 4 5 6 7 8 9 ... ... ... ... ... ... ... ... ... ... 80 (sütun)
C    KARAGECIS NOKTASININ BULUNMASI
     READ (5,20) SAMAS, FIYAT, DEMAL
20   FORMAT (F7.0, 3X, F5.0, 3X, F5.0)
     KARGEN = SAMAS/ (FIYAT — DEMAL)
     WRITE (6,15) KARGEN
15   FORMAT (5X, 'KARAGECIS NOKTASI =', I8,
    ★3X, 'BIRIM MAL')
     STOP
     END
```

Veriler ise 20 nolu FORMAT'a göre aşağıdaki biçimde karta delinecektir (birinci sütundan başlayarak) :

bb4bbbbbbbb2500bbbb1800

Bu program çıktısı da bilgisayar tarafından aşağıdaki gibi yazılacaktır :

bbbbbKARAGECIS NOKTASI = bbbbbb57bbbBIRIMbMAL

Örnek 2 : Bileşik Faizin Hesaplanması

160000 TL., % 50 faiz ile 3 yıllık bir dönem için bankaya yatırılırsa dönem sonunda toplam kaç lira olur?

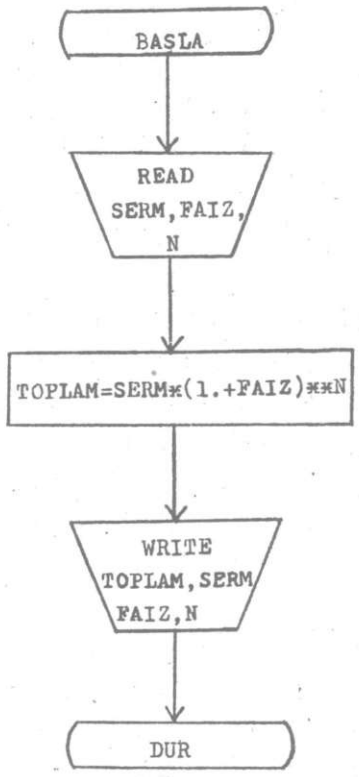
Değişkenler :

SERM	:	Sermaye = 160000 TL.
FAIZ	:	Faiz oranı = 0.50
N	:	Yıl sayısı = 3
TOPLAM	:	Toplam miktar = ?

Yöntem :

$$TOPLAM = SERM \bigstar (1 + FAIZ)^{N}$$

Akış - şeması :



Program :

```
C   BILESIK FAIZ HESABI
      READ (5,10) SERM, FAIZ, N
   10 FORMAT (F6.0/F4.0/I3)
      TOPLAM=SERM*(1.+FAIZ)**N
      WRITE (6,15) SERM, FAIZ, N, TOPLAM
   15 FORMAT (10X, 'BILESIK FAIZ HESABI'/5X,
     1'SERMAYE', 5X, 'FAIZ ORANI', 5X, 'YIL',
     1/5X, F7.0, 8X, F4.2, 8X, I2,/ /,5X,
     1'TOPLAM MIKTAR=', F13.2, 'TL.')
      STOP
      END
```

Bu programa göre veriler,

1. kart 160000
2. kart 0.50
3. kart bb3

biçiminde kartlara delinirse çıktı aşağıdaki gibi olacaktır.

```
bbbbbbbbbBILESIK b FAIZ b HESABI
bbbbbSERMAYE        FAIZ b ORANI bbbbb YIL
bbbbb160000.          0.50bbbbbbbb bb3
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbTOPLAM MIKTAR= bbbb540000.00 TL.
```

Yukarıda verilen basit programlama örneklerinde girdi/çıktı kuralları gösterilmeye çalışılmıştır. Bu nedenle çıktıda boş olarak görülecek yerleri belirtmek için b simgesi sık sık kullanılmıştır. Programlarda dikkat çeken bir nokta birinci programda FORMAT deyiminin iki kartta devam ettiğini belirtmek için devam sütununda (6. sütun) * işaretinin kullanılması, ikinci programda yine FORMAT deyiminin üç kartta devam ettiğini belirtmek için 1 sayısı devam sütununda yazılmıştır.

2.9 Yazılan Programın Bilgisayara Verilmesi

Tek sıra (batch) kullanım yönteminde, delikli kartlara delinmiş bir programın (ki buna kimi zaman programlama çalışmalarında "iş" adı da verilir) bilgisayarda işlenmesi için kartların belli bir sıraya göre düzenlenmesi gerekir. Bilgisayar merkezlerinde kullanıcıya bildirilen bu düzen IBM System/360 için aşağıdaki sırayı izler :

1. Bir programın ilk kartı JOB (iş) kartıdır. Bu kart kullanıcının ya da programcının adını ve bilgisayar muhasebe kayıtları için bilgisayarı kullanma numarasını (user number) belirtir.

2. İkinci kart EXEC kartı adını alır ve hangi derleyicinin kullanacağını belirtir.

3. Üçüncü kart DD kartıdır ve kaynak programın kendisinden sonra geldiğini elirtir.

4. Bu sırada programın (kaynak program) kartları programdaki işlem sıralarına göre dizilir. Yani ilk üç kartı (ki bunlara iş kontrol kartlarıda denilir) programı içeren kartlar izler.

5. Programın sonunda birinci ve ikinci sütunlarında /★ işaretleri olan bir kart yer alır. Bu kart programın bittiğini belirtir.

6. Programın bittiğini belirten kartı ikinci bir DD kartı izler. DD kartı bu kez programın girdileri olan veri kartlarının bitişikte olduğunu bildirir.

7. Bu sırada veri kartları bulunur.

8. Veri kartlarının sonunda yine birinci ve ikinci sütunlarda /★ işaretleri taşıyan bir kart bulunur ve veri kartlarının bittiğini belirtir.

Böylece yukarıdaki biçimde sıralanmış bir kart destesi bilgisayarda işlenmeye hazırdır. Her bilgisayar merkezinde bu sıranın nasıl olduğu ve kontrol kartlarının nasıl delinmesi gerektiğini belirten açıklamalar merkezin kataloglarından ya da yetkililerden sağlanabilir.

Örneğin, daha önceleri ODTÜ'nün IBM 370/145 sisteminde sıralama düzeni (hatadan arındırılmış) bir iş için aşağıdaki gibidir :

```
1. kart    //numarabJOBb, 'ad ve soyad'
2. kart    //bEXECbCSSET2
3. kart    //FORT.SYSINbDDb★
   ⋮
   (program kartları)
   ⋮
.. kart    /★
.. kart    /★GO.SYSINbDDb★
   ⋮
   (veri kartları)
   ⋮
Son kart   /★
```

İlk kartta belirtilen numara, ad ve soyad kısmına programcı (kullanıcı) kendisine ait olanı yazacaktır.

ODTÜ'de şu anda bulunan Burroughs B6900 sisteminde ise kart düzeni şöyledir :

```
1. kart     ?bBEGINbJOBbisim; USER=numara/parola;
2. kart     ?bCOMPILEbprogram adıbFORTRAN;
3. kart     ?bCOMPILERbDATA
:
(program kartları)
:
.. kart     ?bDATA
:
(veri kartları)
:
Son kart  ?bENDbJOB
```

Kontrol kartlarının ilk sütununda bulunan ? işareti multy punch (çoklu delme) olarak aynı sütunda 1,2 ve 3 rakkamlarının delinmesini temsil etmektedir.

Örneğin, birinci örnekte verilen "Kâra Geçiş Noktasının Bulunması" programı iş kontrol kartlarının da eklenmesi ile aşağıdaki biçimde kartlara delinirse program bilgisayara verilmeye (okutulmaya) hazır demektir.

```
?bBEGINbJOBbHASAN; USER=K14067/ELMA;
?bCOMPILEbKARAGECISbFORTRAN
?bCOMPILERbDATA
C  KARA GECIS NOKTASININ BULUNMASI
       READ (5,20) SAMAS, FIYAT, DEMAL
   20  FORMAT (F7.0, 3X, F5.0, 3X, F5.0)
       KARGEN=SAMAS/(FIYAT-DEMAL)
       WRITE (6,15) KARGEN
   15  FORMAT (5X, 'KARAGECIS NOKTASI=', I8,
      ★3X, 'BIRIM MAL')
       STOP
       END
?bDATA
b b 4b bbbbbbb2500bbbb1800
?bEND bJOB
```

Daha öncede belirtildiği gibi programcı adı HASAN kullanıcı numarası K14067, parola ELMA olarak alınmıştır. Program adı da KARAGECIS biçiminde alınmıştır. b harfleri boş sütunu belirtirken ?'de multy punch delmeyi ifade etmektedir. Yukardaki program kartlara olduğu gibi delinir ve bilgisayara verilirse program işlenir ve çıktısı bilgisayar tarafından yazılarak verilir.

2.10 Aktarma Deyimleri

Genel bir kural olarak bir FORTRAN programını oluşturan deyimler programdaki yerlerine göre bir biri ardından sıra ile işlenir. Yani ilk deyimden başlayarak son deyime (END) kadar işlem kontrolü hiç bir yöne saptırılmaz. Ancak kimi zaman çözümü istenilen probleme bağlı olarak, programcı bu sırayı değiştirmek isteyebilir. Örneğin, bir kaç deyimi ya da programın bir bölümünü atlamak ve sonradan tekrar programın ön kısmındaki deyimlere dönülüp bazı deyimlerin tekrarlanması bazılarının atlanması istenebilir. Başka bir anlatımla programcı programın işlem akış yönünü programın başka bölümlerine aktarmak isteyebilir. Programcının bu isteğini aktarma deyimleri karşılar. Aşağıdaki altbölümlerde ayrı ayrı incelenecek olan bu deyimlerin belli başlıları şunlardır :

1. GO TO deyimi
2. Hesaplanmış GO TO deyimi
3. Aritmetik IF deyimi
4. Mantıksal IF deyimi

2.10.1 GO TO Deyimi

Bu deyim program akışının başka bir deyime aktarıldığını yani işlem için başka bir deyime gidileceğini bildirir. Genel kalıbı,

```
GO TO n
```

biçimindedir. Burada n harfi en fazla beş basamaklı bir tamsayıyı temsil etmekte ve program akışının aktarıldığı deyimin numarasını belirtir. Örneğin,

```
GO TO 25
```

gibi. Şimdi aşağıdaki program bölümünü düşünelim :

```
      A=128.0
      B=4.0
      GO TO 25
      A=A+2.0
      C=A*B
   25 D=A/B
      .
      .
      .
```

Bu program bölümünde A değişkeni 128.0, B değişkenide 4.0 değerini aldıktan sonra işlem kontrolü GO TO 25 deyimi ile 25 D=A/B deyimine aktarılmaktadır. Yani GO TO 25 deyimini izleyen A=A+2.0 ve C=A★B deyimleri işlenmemektedir.

Burada ki GO TO 25 deyimi herhangi bir koşulu gerektirmeden doğrudan aktarma işini sağladığı için kimi zaman koşulsuz GO TO deyimi olarak da adlandırılır.

2.10.2 Hesaplanmış GO TO Deyimi

Bu deyim aktarma işini belli koşullara göre yapmakta ve

GO TO (n_1,n_2,n_3,.....,n_j),i

genel kalıbı ile ifade edilmektedir. Burada;

n_j : deyim numaralarını,

i : harfide bir tamsayı değişkenini göstermektedir.

Bu deyim şunu ifade etmektedir: eğer i'nin değeri 1 ise n_1 nolu, 2 ise n_2 nolu, 3 ise n_3 nolu,....., ve eğer i'nin değeri j ise n_j nolu deyime kontrol aktarılacak ve bu deyimlerin belirteceği işlem yapılacaktır. Örneğin;

```
      :
      GO TO (10, 45, 25, 30, 15), NYIL
10    AYLIK=AYLIK+300.0
      :
45    AYLIK=AYLIK+500.0
      :
25    AYLIK=AYLIK+700.0
      :
30    AYLIK=AYLIK+800.0
      :
15    AYLIK=AYLIK+1000.0
      :
      :
```

biçiminde yazılan bir program bölümünü işçi aylıklarının hesaplanmasında kullandığımızı varsayalım.

NYIL değişkeni işçinin kaç yıl çalıştığını AYLIK değişkeni de işçinin aylığını göstersin. Buna göre eğer :

NYIL=1 ise 10 noul deyime göre işçinin aylığına 300
NYIL=2 ise 45 nolu deyime göre işçinin aylığına 500
NYIL=3 ise 25 nolu deyime göre işçinin aylığına 700
NYIL=4 ise 30 nolu deyime göre işçinin aylığına 800
NYIL=5 ise 15 nolu deyime göre işçinin aylığına 1000

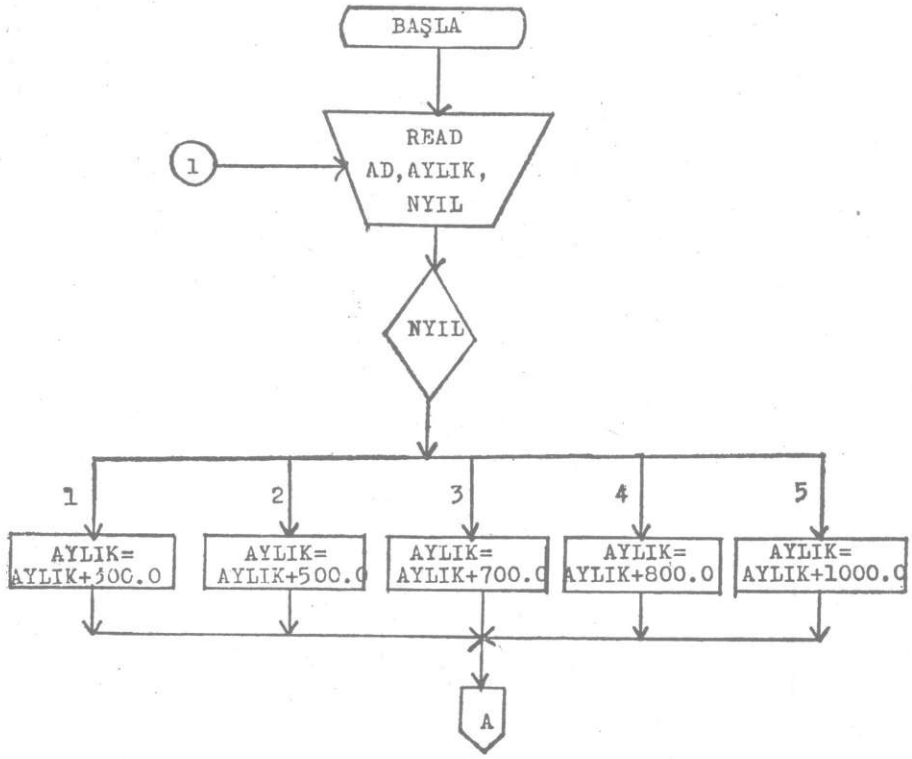
lira eklenecektir.

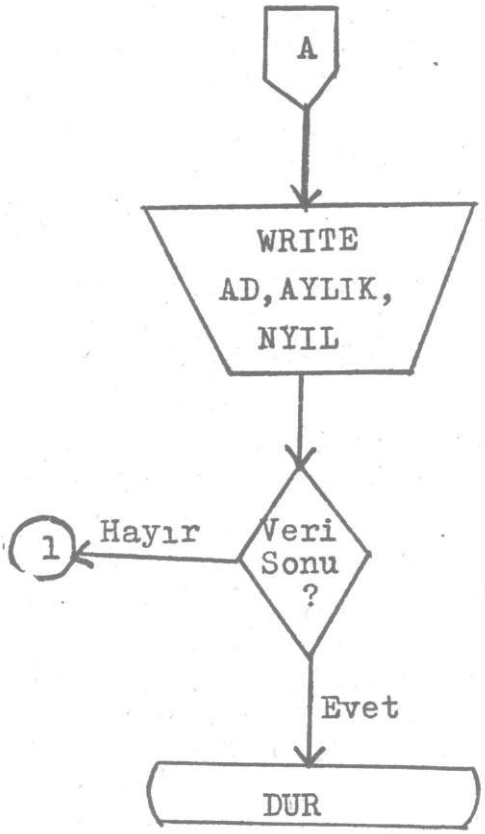
Görüldüğü gibi hangi deyimin işleneceği yani aylığa ne kadar ekleme yapılacağı NYIL değişkeninin alacağı değere bağlıdır. Bu nedenle bu deyim koşullu bir ayırma ya da aktarma işlevi görmektedir.

Örnek : Ücret Düzenleme

Yukarıda verilen ücretlere ekleme (zam) yapma örneğini tamamlayalım. Kullanılan değişkenler aynıdır. Yalnız işçi adlarını belirten bir AD değişkeni alalım. Böylece programın amacı işçilerin çalışma yıllarına göre belirtilen eklemeleri yapmak ve belirlenen yeni aylıkları çıktıda yazdırmaktır. Örnekte 5 yıldan fazla çalışan işçi olmadığı varsayılmıştır.

Akış - şeması :





Akış-şemasında görüldüğü gibi NYIL değişkeninin değerine göre beş ayrı seçenekten biri seçilecek ve aylık düzenleme yapılıp sonuç yazılacaktır. Sonra verilerin sonunun gelip gelmediği kontrol edilecektir. Eğer verilerin sonu gelmemişse yani veriler bitmemişse kontrol tekrar baştaki READ deyimine aktarılacak ve yeni veriler okunarak ücret düzenleme için NYIL değişkenine göre gerekli işlemler yapılacaktır. Verilerin sonu geldiğinde kontrol STOP deyimine geçecek ve program bitmiş olacaktır. Akış-şemasına dayalı olarak aşağıdaki gibi yazılabilir.

```
C   UCRET DUZENLEME PROGRAMI
C   BAŞLIK YAZDIRMA
        WRITE (6,10)
   10 FORMAT (1H1, 5X, 'UCRET DUZENLEME', / /, 5X,
     1'ADI VE SOYADI', 5X, 'YENI AYLIGI', 5X, 'YIL')
    1 READ (5,15, END=80) AD1, AD2, AD3, AD4, AYLIK,
     1NYIL
```

```
15 FORMAT (4A4, 5X, F8.0, 5X, I2)
   GO TO (20, 25, 30, 35, 40), NYIL
20 AYLIK=AYLIK+300.0
   GO TO 45
25 AYLIK=AYLIK+500.0
   GO TO 45
30 AYLIK=AYLIK+700.0
   GO TO 45
35 AYLIK=AYLIK+800.0
   GO TO 45
40 AYLIK=AYLIK+1000.0
45 WRITE (6,50) AD1, AD2, AD3, AD4, AYLIK, NYIL
50 FORMAT (5X, 4A4, 2X, F8.2, 7X, I2)
   GO TO 1
80 STOP
   END
```

Örnek veriler ve 15 nolu FORMAT'a göre kartlara delinmesi :

```
1. kart   HASANbYILMAZbbbbbbbbb15462.80bbbbbb4
2. kart   HUSEYIN BAYSAL          17250.50      1
3. kart   MAHMUT SARI             13000.00      3
```

Programda görüldüğü gibi ilk WRITE deyimi yalnızca başlıkları yazmak için kullanılmıştır. Bu nedenle bir değişken listesi yoktur. 1 READ deyimi gerekli verileri okutmak için yazılmıştır. Bu deyimdeki END=80 ifadesi okunacak verilerin sonu geldiğinde (bittiğinde) kontrolu 80 nolu deyime aktarmak için kullanılmıştır. 80 nolu deyim ise STOP deyimidir. Böylece programın veriler bittikten sonra tekrarı önlenmiştir.

Yukarıda verilen örnek verilere göre çıktı (yeni sayfada) :

```
1. satır   bbbbbUCRETbDUZENLEME
2. satır
3. satır          ADI VE SOYADI        YENI AYLIGI    YIL
4. satır          HASAN YILMAZ         16262.80       4
5. satır          HUSEYIN BAYSAL       17550.50       1
6. satır          MAHMUT SARI          13700.00       3
```

2.10.3 Aritmetik IF Deyimi

Bu deyimde koşullu bir deyimdir ve üç yönlü bir aktarma işlevi görür. Deyimin genel kalıbı,

IF (e) n_1, n_2, n_3

biçimindedir. Burada;

n_1, n_2, n_3 : deyim numaralarını,

e : bir aritmetik ifadeyi ya da karmaşık (kompleks) olmayan bir gerçel ya da tamsayı değişkenini temsil etmektedir.

IF : "Eğer,... ise" anlamına gelen bir sözcüktür.

Böylece deyim aşağıdaki ifadeyi belirtmektedir :

Eğer (e)'nin değeri negatif ($e < 0$) ise kontrol n_1 nolu,
eğer (e)'nin değeri sıfır ($e = 0$) ise kontrol n_2 nolu, ve
eğer (e)'nin değeri pozitif ($e > 0$) ise kontrol n_3 nolu

deyime aktarılır. Örneğin;

```
      .
      .
      .
      IF (A-B*2.0) 5, 10, 20
    5 C=A*D+E
      .
      .
   20 X=A+D
      .
      .
   10 B=(A+E)*3.0
      .
      .
```

Bu program bölümünde eğer (A—B★2.0) ifadesinin sonucu negatif ise kontrol 5 nolu deyime, sıfır ise 10 nolu deyime, pozitif ise 20 nolu deyime aktarılır ve deyimlerin belirttiği gerekli işlemler yapılır.

Bir başka örnek olarak aşağıdaki ifadeleri alalım :

```
      .
      .
      IF (GELIR-GIDER) 10, 10, 20
   10 VERGİ=0.0
      .
      .
   20 VERGI=(GELIR-GIDER)*A
      .
```

Bu program bölümünde GELIR ve GIDER değişkenleri arasındaki fark negatif ya da sıfır ise kontrol 10 nolu deyime aktarılmakta ve VERGI değişkeni sıfır değerini almaktadır. Eğer fark pozitif ise kontrol

20 nolu deyime aktarılmakta ve GELİR ile GIDER değişkenleri arasındaki fark A değişkeninin değeri ile çarpılmaktadır, yani A oranında VEGI alınmaktadır.

Yukarıdaki program bölümü aşağıdaki biçimde de yazılabilir:

```
      REAL KAR
      :
      KAR=GELIR-GİDER
      IF (KAR) 10, 10, 20
   10 VERGİ=0.0
      :
      :
   20 VERGI=KAR*A
      :
      :
```

Verilen örneklerde görüldüğü gibi ayraç içindeki ifade bir aritmetik ifade olacağı gibi bir gerçel ya da tamsayılı değişken de olabilir. Ayrıca ayraçtan sonraki deyim numaralarıda ayrı olmak zorunda değildirler. KAR tamsayılı değişkeni REAL tip bildirme deyimi ile gerçel bir değişken haline dönüştürülmüştür.

2.10.4 Mantıksal IF Deyimi

İki yönlü bir aktarma işlevi gören bu deyim aşağıdaki genel kalıp ile ifade edilir :

```
IF (e) S
```

Burada;

e : mantıksal bir ifadeyi,

S : işlenebilir bir deyimi temsil etmektedir. Ancak S işlenebilir deyimi bir IF ya da DO deyimi olmamalıdır.

Buna göre yukarıdaki deyimin anlamı : (e) mantıksal ifadesi doğru ise S deyiminin belirttiği işlem yapılacaktır.

Eğer (e) mantıksal ifadesi doğru değilse yani yanlış ise IF deyiminden hemen sonra gelen deyim işlenecektir. Yani bu durumda S deyiminin belirttiği işlem yapılmayacaktır.

Bir mantıksal ifade, ilişki belirleyicileri aracılığı ile birleştirilen aynı türden iki aritmetik ifade ya da değişkenden (kompleks değişkenler hariç) oluşur. Kendilerinden önce ve sonra birer nokta (.) bulunan ilişki belirleyicileri şunlardır :

Adı	Anlamı
.GT.	.... büyüktür (Greater Than)
.GE.	.... büyük ya da eşittir (Greater than or Equal to)
.LT.	.... küçük (Less Than)
.LE.	.... küçük ya da eşittir (Less than or Equal to)
.EQ.	.... eşittir (EQual)
.NE.	.... eşit değildir (Not Equal)

Örnek :

```
      ⋮
      IF (A—B.LT.C) B=A+C
      B=B+C★2.0
      ⋮
      IF (A.GE.C) GO TO 25
      B=A+C
      ⋮
   25 B=A★★2—C
      ⋮
```

Yukarıdaki ilk IF deyiminde eğer A—B'nin sonucu C'den küçük ise B=A+C işlemi yapılacaktır. Eğer bu ifade doğru değilse, yani A—B'nin sonucu C'den küçük değilse (büyük ya da eşit ise) B=B+C★2.0 işlemi yapılacaktır. İkinci IF deyiminde eğer A değişkeninin değeri C değişkeninin değerinden büyük ya da eşit ise işlem için GO TO 25 deyiminin gereği olarak 25 nolu deyime (B=A★★2—C) kontrol aktarılacak ve bu deyim işlenecektir. Eğer mantıksal ifade (A.GE.C) doğru değilse yani A'nın değeri C'den büyük ya da eşit değilse (küçük ise) B=A+C işlemi yapılacaktır.

Aynı şekilde, aritmetik IF ile ilgili olarak verilen örnekte,

```
      REAL KAR
      ⋮
      KAR=GELIR—GIDER
      IF (KAR) 10, 10, 20
   10 VERGI=0.0
      ⋮
   20 VERGI=KAR★A
      ⋮
```

deyimleri mantıksal IF deyimi ile aşağıdaki gibi yazılabilir.

```
      REAL KAR
      .
      .
      .
      KAR=GELIR-GİDER
      IF (KAR. LE.0.0) GO TO 10
      VERGI=KAR★A
      GO TO 5
   10 VERGI=0.0
    5 ......
      .
      .
      .
```

Yukarıdaki deyimlere göre eğer KAR sıfırdan (0.0) küçük ya da sıfıra eşitse vergi alınmayacaktır, yani GO TO 10 deyimine göre 10 nolu deyime geçilip VERGI=0.0 yapılacaktır. Aksi durumda yani KAR sıfırdan büyük olma durumunda VERGI=KAR★A işlemine göre vergi alınacaktır. Böylece iki seçenek söz konusu olmaktadır : ya vergi alınacak ya da alınmayacaktır. Hangi seçeneğin uygulanacağı ise KAR değişkeninin sıfır ile karşılaştırmasına bağlıdır.

GO TO 5 deyimi ise vergi alındıktan sonra VERGI=0.0 işlemini yaptırmamak yani bu deyimi atlamak için kullanılmıştır.

Aynı ifadelerin;

```
      REAL KAR
      .
      .
      .
      KAR=GELIR-GIDER
      IF (KAR.LE.0.0) VERGI=0.0
      VERGI=KAR★A
      .
      .
      .
```

olarak yazıldığını varsayalım. Hatalı olarak yazılan bu durumda eğer mantıksal ifade doğru değilse, VERGI=KAR★A işlemine göre vergi hesaplanacak ve ondan sonra gelecek olan deyimlerin işlemleri yapılacaktır. Aksi durumda yani mantıksal ifadenin (KAR.LE.0.0) doğru olması durumunda (KAR sıfırdan küçük ya da eşit ise) VERGİ=0.0 işlemi yapılacak sonra yine VERGI=KAR★A işlemi yapılacaktır. Çünkü VERGI=KAR★A deyimi VERGI=0.0 işleminden sonra yer almıştır.

Böylece vergi hesaplanması işlemi iki kez yapılmıştır. Görüldüğü gibi (KAR.LE.0.0) mantıksal ifadesinin doğru olması durumunda yani kârın olmadığı zaman bile vergi hesaplanacaktır. Bu nedenle mantıksal IF deyiminin kullanıldığı durumlarda böyle noktaların göz önünde bulundurulması gerekir. Yani arzu edilmeyen işlemlerden sakınmak gerekir.

Manktıksal ifadeler,.AND.,.OR. ve .NOT. mantıksal belirleyiciler kullanılarak birleştirilebilir. (.AND.) iki ayrı koşulun aynı anda olmasını, (.OR.) ise iki ayrı koşuldan birisinin olması gerektiğini bildirir. (.NOT.) ise bir olumsuzluk durumunu bildirir. Örneğin;

```
      .
      .
      .
      IF (A.GT.B.AND.A.LE.C) Y=A+B/2.0
      X=A/B
      .
      IF (A.EQ.Y.OR.X.LT.3.0) GO TO 40
      C=C+B/2.0
      .
   40 C=C+B
      .
      .
      .
```

deyimlerinde birinci IF deyiminde eğer A'nın değeri B'den büyük ve C'den küçük ya da eşit ise Y=A+B/2.0 işlemi yapılacaktır. Daha sonrada X=A/B işlemi yapılacaktır. Aksi durumda ise, yani bu koşulların aynı anda sağlanamaması durumunda ise yalnızca X=A/B işlemi yapılacaktır. İkinci IF deyiminde ise, eğer A değişkeninin değeri Y'nin değerine eşit ya da X değişkeninin değeri 3.0'den küçük ise GO TO 40 deyiminin gereği olarak kontrol 40 nolu deyime aktarılacak ve C=C+B işlemi yapılacaktır. Aksi durumda yani bu iki koşuldan her hangi birinin karşılanmaması durumunda ise C=C+B/2.0 işlemi yapılacaktır.

2.11 Örnek Programlar

Örnek 1 :

1'den N'ye kadar olan sayıların toplamının bulunması :

Toplam sonucu IT ile gösterirsek yapılacak olan işlem, IT=1+2+3+4+.....+N şeklinde belirtilebilir. Örneğin, N=10 olarak verilmiş olsun. Yapılacak işlem birer birer artan ve 10'a kadar devam eden dizinin toplamını bulmaktır.

Toplam sonucu IT ile gösterdiğimizde IT'nin ilk başlangıç değeri sıfırdır. Çünkü hiç bir toplama işlemi yapılmamıştır. Ondan sonra IT değeri her toplama aşamasında değişecekti. Diğer yandan birer birer artan diziyi temsil eden değişkeni de K ile gösterelim. Her toplama aşamasından sonra K değişkenini 1 artıralım. Buna göre akış-şeması ve program bitişik sayfadaki gibi yazılabilir. READ deyimi yerine, N'nin değerinin 10 olduğu N=10 atama deyimi ile bildirilmiştir.

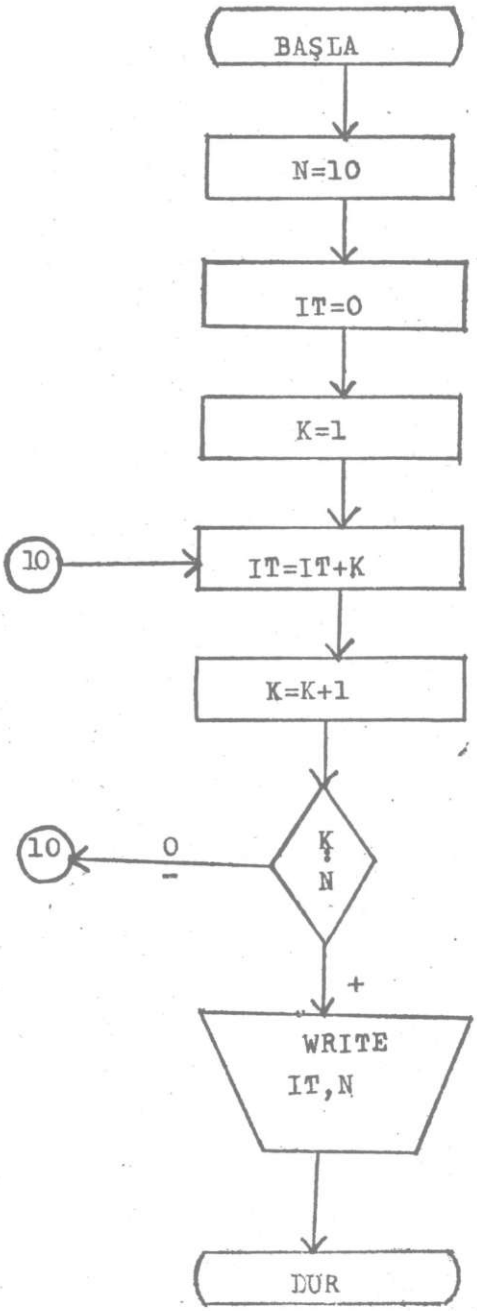
Akış-şemasında görüleceği gibi;

IT=0
K=1

atama deyimleri ile bu iki değişken için başlangıç değerleri oluşturulmuştur. Her toplama aşamasında 1 artan K değişkeninin başlangıç değeri de K=0 olarak oluşturulabilirdi. Ancak bu değişken her aşamada bir arttığı için K=1 olarak belirlenmiştir.

K değişkeni birer birer artan bir sayma işlevi görmektedir. Yani 1'den başlayıp N değerine kadar her aşamada bir artacaktır. K değişkeni değerinin N değerine eşit olup olmadığını kontrol etmek için aritmetik IF deyimi kullanılacaktır. Kontrol bir karar verme noktası olduğu için aritmetik IF deyimi akış şemasında karar noktasını belirten eşkenar dörtgen şeklinde belirtilmiştir. Başka bir anlatımla bu noktada K : N değişkenleri karşılaştırılmaktadır. Aritmetik IF deyimi ile (K—N) farkı kontrol edilmektedir. (K—N) farkı pozitif olduğu zaman yani K değişkeninin değeri N değişkeninin değerinden büyük olduğu an K'nın sayma işlevi bitmekte ve o ana kadar hesaplanan toplam (IT) yazdırılacaktır. Yazdırma işleminde K'nın değeri istenilmemiştir. Eğer istenilirse K=11 olarak belirtilecektir. Ancak akış-şemasında görüldüğü gibi K değeri N'den büyük olduğu zaman bu değer toplama işlemine eklenmemektedir. Aritmetik IF deyiminin kontrol işlevini daha açık görebilmek için bu basit örneğin yararlı olacağı düşüncesi ile programdan sonra sayısal değerleri ile belirtilecektir.

Akış - Şeması



Program :

```
    N=10
    IT=0
    K=1
10  IT=IT+K
    K=K+1
    IF(K—N)  10, 10, 20
20  WRITE  (6,15)  IT, N
15  FORMAT  (5X, I7, 5X, I3)
    STOP
    END
```

Verilen programın akışı aşağıdaki aşamaları izler :

1. N=10 değerini alır.
2. IT=0 değerini alır.
3. K=1 olur.
4. IT=IT+K deyimine göre, IT=0+1=1 olur.
5. K=K+1 deyimine göre, K=1+1=2 olur.
6. K ile N karşılaştırılır. Eğer (K—N)'nin sonucu negatif ya da sıfır ise 10 nolu deyime aktarma yapılır. Eğer (K—N) sonucu pozitif ise WRITE deyimi işlenir. Bu aşamada K—N=2—10= —8 olduğundan 10 nolu deyime dönülür ve buna göre IT=IT+K=1+2=3 bulunur.
7. K=K+1 deyimine göre K bir artırılır. Yani, K=2+1=3 olur.
8. Tekrar K ve N karşılaştırılır. K—N=3—10=—7 olduğundan 10 nolu deyime tekrar dönülür.
9. IT=IT+K ifadesine göre IT=IT+K=3+3=6 bulunur.
10. K=K+1 deyimine göre K yine 1 artılır ve K=3+1=4 bulunur.
11. Yine K ve N karşılaştırılır ve yukarıdaki gibi işlemler devam eder.

Son aşamada K=10 olunca K ile N tekrar karşılaştırılır ve (K—N) sonucu sıfır olduğu için yine 10 nolu deyime dönülür. Çünkü K=10 değeri IT=IT+K toplama işlemine konulmamıştır. 10 nolu deyimde K'nın son değeri olan 10, toplam IT değerine eklendikten sonra K=K+1 deyimine göre K=10+1=11 bulunur. Tekrar K ile N karşılaştırılır. Ancak bu kez K—N=11—10=1 olduğu için 20 nolu deyime kontrol aktarılır. 20 nolu deyim ise programın işlem sonuçlarını yazdıracaktır. Böylece program işleyişi sona erecektir. Yani STOP ve END deyimlerinin gereği yapılacaktır.

Yukarıda belirtilen program biraz değişik olarak aşağıdaki biçimde de yazılabilir.

```
    N=10
    IT=0
    K=0
10  K=K+1
    IT=IT+K
    IF (K—N) 10, 20, 20
20  WRITE (6,15) IT, N
```

```
15 FORMAT (5X, I7, 5X, I3)
   STOP
   END
```

Bu programda farklılık K değişkeninin toplama işlemine konulmadan önce 1 artırılmasıdır. Kuşkusuz bu da aritmetik IF deyiminde bir uyarlama gerektirmektedir. Yani (K—N) sonucu sıfır olunca programın akışı WRITE deyimine aktarılmaktadır. Böylece K'nın 11 değerini alması ve IT toplamına eklenmesi önlenmiş oluyor.

Bu programların vereceği sonuç $IT=\frac{N(N+1)}{2}$ formülü ile de bulunur. Ancak aritmetik IF deyimine bir örnek vermek ve birer birer saydırma işleminin nasıl yapılabileceğini açıklamak için diğer yöntem burada ele alınmıştır.

Örnek 2 : Yıl Basamakları Toplamı (Azalan Bakiyeler) Yöntemi ile Aşınmanın Hesaplanması

Bu yöntemde sermaye malının (makina, bina, v.b) değeri her yıl değişen bir oranda düşürülür. Bu yöntemle yıllık aşınmanın nasıl hesaplandığını bir örnekle açıklayalım.

Örneğin, değeri 150 bin lira ,ömrüde 5 yıl olan bir makinanın yıllık aşınma payları nedir? Makinanın değerini DEGER, ömrünü N ve hesaplanacak olan yıllık aşınma paylarını da ASINMA değişkenleri ile gösterelim. Yıl basamakları toplamını (yani, 1+2+3+4+5=15) bulmak için de N(N+1)/2 formülünü kullanırsak yöntem aşağıdaki aşamaları izler :

1. yıl $ASINMA=\frac{DEGER}{N(N+1)/2}\ N = \frac{150000}{5(5+1)/2}\ 5=50000$ TL.

2. yıl $ASINMA=\frac{DEGER}{N(N+1)/2}\ (N-1) = \frac{150000}{5(5+1)/2}\ (5-1)=$ 40000 TL.

3. yıl $ASINMA=\frac{DEGER}{N(N+1)/2}\ (N-2) = \frac{150000}{5(5+1)/2}\ (5-2)=$ 30000 TL.

⋮

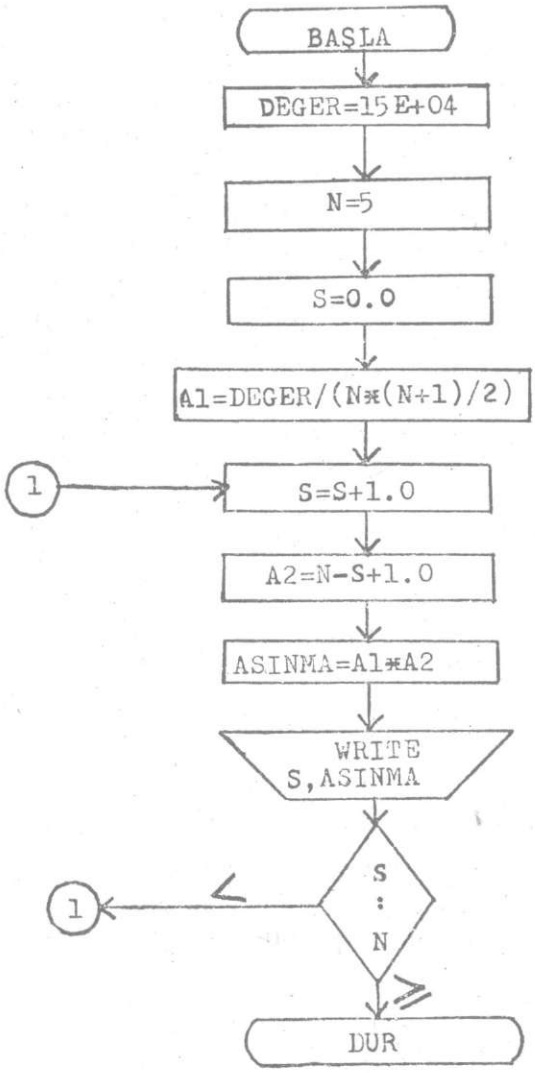
5. yıl $ASINMA=\frac{DEGER}{N(N+1)/2}\ (N-4) = \frac{150000}{5(5+1)/2}\ .(5-4)=$ 10000 TL.

Bu yöntemi her yıl için genelleştirirsek aşağıdaki biçimde yazabiliriz.

$$ASINMA=\frac{DEGER}{N(N+1)/2}\ (N-S)$$

S simgesi yukarıdaki örnekte görüldüğü gibi değeri sıfırdan (N—1)'e kadar birer birer artan bir değişkeni temsil etmektedir. Verilen örnekte S'nin en son değeri 4 olmaktadır.

Bu açıklamalardan sonra şimdi aynı örneği bilgisayar ile yapalım. Genel formülde görüldüğü gibi yıllık aşınmanın hesaplanması iki kısımdan oluşmaktadır. Birinci kısım $\frac{DEGER}{N(N+1)/2}$ dir. Bunu A1 değişkeni ile gösterelim. İkinci kısım, yani (N—S) ifadesini de A2 ile belirtirsek ASINMA=A1★A2 biçiminde gösterilebilir. Buna göre akış-şeması ve program aşağıdaki gibi yazılabilir.



Program :

```
C  YIL BASAMAK YONTEMINE GORE ASINMANIN
C  HESAPLANMASI
       REAL N
       DEGER=15E+04
       N=5.0
       S=0.0
       WRITE (6,15) DEGER, N
   15  FORMAT (1H1,/ /, 5X, 'SERMAYE DEGERI=',
     ★F9.0, 'TL.',/ /, 5X, 'SERMAYENIN OMRU=',
     ★F3.0, 'YIL', / /, 5X, 'YIL', 5X, 'ASINMA PAYI')
       A1=DEGER/(N★(N+1.0)/2.0)
   25  S=S+1.0
       A2=N-S+1.0
       ASINMA=A1★A2
       WRITE (6,20) S, ASINMA
   20  FORMAT (5X, F3.0, 5X, F11.2)
       IF (S.LT.N) GO TO 25
       STOP
       END
```

Buna göre çıktı yeni bir sayfada aşağıdaki gibi olacaktır.

bbbbbSERMAYEbDEĞERI=bb150000. TL.

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

SERMAYENIN OMRU= 5. YIL

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

YIL	ASINMA PAYI
1.	50000.00
2.	40000.00
3.	30000.00
4.	20000.00
5.	10000.00

Yukarıda verilen program örneğinin bazı özellikleri şunlardır :

a) REAL N tip bildirme deyimi ile bir tamsayı değişkeni olan N gerçel bir değişken olarak işleme konulmuştur.

b) WRITE (6,15) deyimi ile programın başında bir defa yazılması arzu edilen bir ileti yazılmış oluyor.

c) A1=DEĞER/(N★(N+1.0)/2.0) ifadesi ile aşınmanın ilk kısmı hesaplanıyor. Bu işlemin bir kez hesaplanması yeterli olduğundan IF deyimi ile yapılacak olan tekrarlamaların dışında bırakılıyor. Böylece her yıl için tekrar tekrar hesaplanması önlenmiş olmaktadır.

d) 25 S=S+1.0 deyimi ile S'nin birer birer artırılması sağlanıyor. S başlangıçta sıfır (S=0.0) yapıldığı için işlem sırası S=S+1.0 deyimine geldikçe her seferinde S'nin değeri 1 artacaktır. Böylece bu değişken yılları birer birer saymak için kullanılmaktadır.

e) A2=N−S+1.0 deyimi ile her yıl 1 eksilen ve aşınmanın ikinci kısmı olan A2'nin değeri hesaplanıyor. Çözüm yöntemine göre 1. yılda yani S=1.0 iken A2=N olmalı ve 5. yılda yani S=5.0 iken A2=1.0 olmalıdır. Bunu sağlamak için A2 ifadesinin sonuna 1.0 eklenmiştir. Böylece S=1.0 iken A=5.0−1.0+1.0=5.0 (yani N'nin değeri) ve S=5.0 iken A2=5.0−5.0+1.0=1.0 olur. Bu ise örnekteki (N−4) ya da (5−4) ile aynı anlamdadır. Çünkü sonuç 1 olacaktır.

f) AŞINMA=A1★A2 deyimi ile değerleri önceden hesaplanmış olan A1 ve A2'ye göre yıllık aşınmalar hesaplanıyor.

g) WRITE (6,20) deyimi ile her yıl için hesaplanan yıllık aşınmanın sayısal değeri yılı ile birlikte her yıl için ayrı ayrı yazılıyor. Bunu sağlamak için de IF deyimi ile yapılan tekrarların içine alınmıştır.

h) IF deyimi ile de tekrarlama ve programın işleyişinin bitirilmesi sağlanmaktadır. Bu deyime göre S'nin N'den küçük olan değerleri için kontrol 25 S=S+1.0 deyimine aktarılıp S bir artırılarak işlem tekrarlanmaktadır. S'nin N'ye eşit ya da büyük değerleri için işlem yapılmamaktadır. Yani program sona ermektedir.

Örnek 3. Genişliğin (ranjın) Hesaplanması

İstatistiksel anlamda genişlik (ranj), en büyük değer ile en küçük değer arasındaki farktır. Örneğin, bir işletmenin belli bir zaman dönemi ya da belli bölgelerdeki satış miktarlarının en büyüğü ile en küçüğü arasındaki fark işletmenin satış dağılımının genişliğini belirtir. Bunun için işletmenin satış miktarlarını gösteren liste ya da tabloya bir göz atmak, en büyük ile en küçük değerlerin bulunması için yeterlidir.

Ancak bilgisayar ile en büyük ve en küçük değerlerin bulunması programlama açısından bir kaç önemli noktanın göz önünde bulundurulmasını gerektirmektedir. Satış miktarlarının genişliğini bulmak için hazırlayacağımız programda kullanılan değişkenler :

SATIŞ : Satış miktarları.

EBÜYÜK : En büyük satış miktarı.

EKÜCUK : En küçük satış miktarı.

GENLIK : Genişlik.

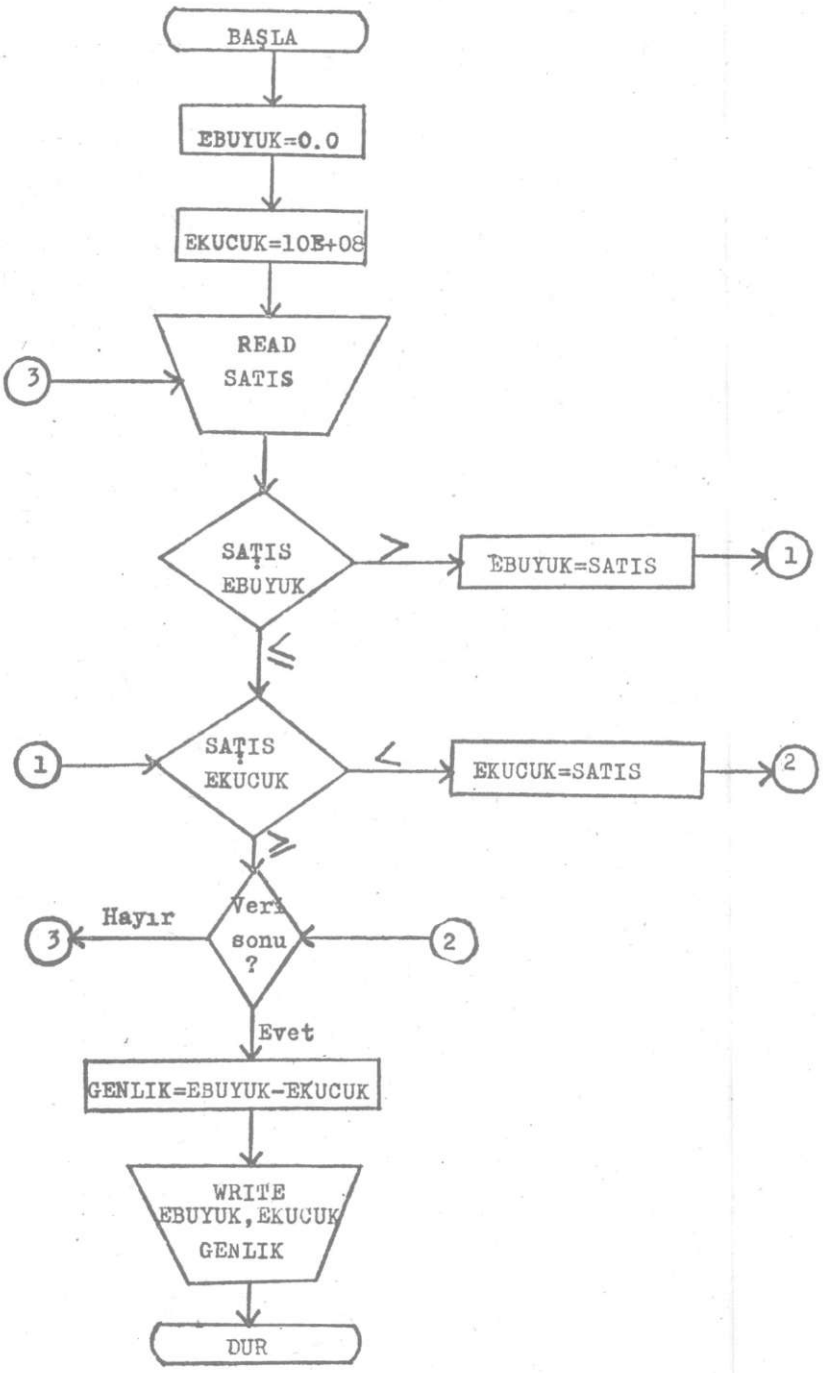
Bilgisayarda en büyük ve en küçük değerleri bulmak için satış miktarları tek tek karşılaştıracaktır. Karşılaştırma yapılırken ilk satış miktarı temel olarak alınacaktır. Bu değerin en küçük ya da en büyük olduğu bilinmemektedir. Bu nedenle ilk satış miktarının hem EBÜYUK hemde EKUÇUK değişkenlerine verilmesi (atanması) gerekir. Çünkü ilk satış miktarı en büyük olabileceği gibi en küçük de olabilir, hiç olmayabilir de. Bu atamaları sağlamak için EBUYUK=0.0 yapılırsa ve EKÜÇUK değişkenine de olası satış miktarlarından daha büyük, örneğin 10E+08 gibi, bir değer verirsek ilk okunacak satış miktarının değerini bitişik sayfada verilen akış-şemasında görüldüğü gibi her iki değişkene aynı anda atayabiliriz.

Akış-şemasında görüldüğü gibi okunan ilk satış miktarı eğer sıfırdan büyük ise EBUYUK değişkeni bu değeri alacaktır. Satış miktarlarının negatif olması söz konusu olmadığına göre ilk satış miktarı EBUYUK değişkene atanacaktır. Aynı şekilde ilk satış miktarının değeri ne olursa olsun EKUCÜK değişkenine de verilecektir. Çünkü EKUCUK değişkeninin başlangıç değeri olası satış miktarlarının hepsinden daha büyük tutulmuştur. Böylece ilk satış miktarı 10E+08 değerinden küçük olacağı için ilk aşamada EKKUCUK değişkeni ilk satış miktarının değerini alacak ve 10E+08 değeri silinecektir.

Yukarıdaki açıklamalara göre programın işleyişinin ilk aşamasında EBUYUK değişkeninin değeri ya sıfır ya da sıfırdan büyük olabilecek olan ilk okunan satış miktarının değeridir. EKÜCUK değişkeninin değeri de ilk okunan satış miktarının değeridir. Bundan sonraki aşamalarda okunacak olan satış (SATIS) miktarları bu değerler ile karşılaştırılacaktır. Eğer SATIS değeri EBÜYUK değerinden büyük ise EBUYUK=SATIS olacaktır. Aksi durumda SATIS değeri EKUCUK değişkeninin değeri ile karşılaştırılacaktır. Bu karşılaştırmada eğer SATIS değeri EKÜCÜK değerinden küçük ise EKÜCUK=SATIS olacaktır.

Bir satış elemanı için büyük ve küçük karşılaştırılması ve gerekli atamalar yapıldıktan sonra sonraki her satış elemanı için aynı işlemleri yapmak için bilgisayar işlem kontrolü GO TO 3 deyimi ile baştaki READ deyimine aktarılmaktadır. Böylece tekrar bir okuma işlemi ya-

Akış - Şeması


BAŞLA
EBUYUK=0.0
EKUCUK=10E+08
READ
SATIS
3
SATIS
EBUYUK
>
EBUYUK=SATIS
1
=<
1
SATIS
EKUCUK
<
EKUCUK=SATIS
2
=>
Veri
sonu
?
Hayır
3
2
Evet
GENLIK=EBUYUK-EKUCUK
WRITE
EBUYUK, EKUCUK
GENLIK
DUR

pılmakta ve o satış elemanı içinde gerekli karşılaştırma ve atamalar yapılmaktadır. Bu işlemler okunacak satış elemanı (yani veri) kalmayıncaya kadar devam etmektedir. Veriler bitince READ deyimi içinde belirtilecek bir END=100 ifadesi ile kontrol GENLIK=EBUYUK—EKUCUK deyimine aktarılacak ve genişlik hesaplanacaktır. Sonra çıktı yazılacak ve program işleyişi bitecektir. Akış şemasına dayalı olarak program aşağıdaki gibi yazılabilir.

Program :

```
C  SATIS DAGILIMI GENISLIGINI
C  HESAPLAMA
       EBUYUK=0.0
       EKUCUK=10E+08
    3  READ (5,15, END=100) SATIS
   15  FORMAT (F8.2)
       IF (SATIS.GT.EBUYUK) EBUYUK=SATIS
       IF (SATIS.LT.EKUCUK) EKUCUK=SATIS
       GO TO 3
  100  GENLIK=EBUYUK—EKUCUK
       WRITE (6,25) EBUYUK, EKUCUK, GENLIK
   25  FORMAT (5X, 'EN BUYUK SATIS MIKTARI=', F8.2,/ /
      ★5X, 'EN KUCUK SATIS MIKTARI=', F8.2,/ /,
      ★5X, 'SATIS DAGILIMI GENISLIGI=', F8.2)
       STOP
       END
```

2.12 Döngü ve Dizinli Değişkenler

2.12.1 DO Deyimi ve Döngü Kavramı

DO deyimi güçlü bir kontrol deyimidir. Bir programın istenilen bölümlerinin istenilen sayıda peş peşe tekrarlanıp işlenmesini sağlar. Programın istenilen bölümünde bilgisayar kontrolü, istenilen sayıda programın o bölümünü oluşturan deyimlerin işlenmesini sağladığı için DO deyiminin tekrarladığı işlem alanına döngü adı verilir.

DO deyiminin genel kalıbı :

DO n i=m_1,m_2,m_3

biçimindedir. Burada;

n : Döngünün kapsadığı alanın içindeki en son deyimin numarasını temsil eden bir tamsayıdır.

i : Dizinli (indisli) olmayan tamsayılı bir değişkeni temsil etmektedir.

m_1 : i değişkeninin başlangıç değeri olup 1 ya da 1'den büyük bir tamsayı sabitini temsil eder.

m_2 : i değişkeninin alacağı en son değer olup m_1'den büyük ya da eşit bir tamsayı sabitini belirtir. $m_2=m_1$ durumunda döngü içindeki işlemler bir kez işlenir.

m_3 : Döngü alanının her tekrarında i değişkeninin alacağı artış değerini göstermektedir. m_3 değeri 1 ya da 1'den büyük bir tamsayı olmalıdır. m_3 belirtilmediği zaman değeri 1 olarak varsayılır.

Örneğin;

```
.....→ DO 20 NL=1, 15, 1
       :
       :
       :
   20 .....
```

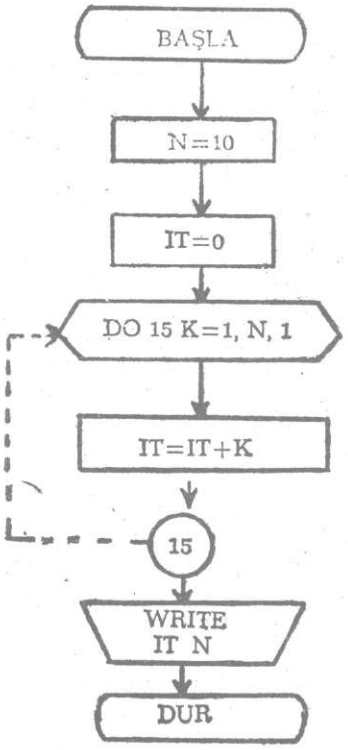
biçimindeki bir döngü çerçevesine göre programda işlem sırası DO deyimine gelince i değişkeni (NL) m_1 değerini (1) alacak ve n nolu (20) deyime kadar (budeyim dahil) olan işlemler yapılacaktır. Sonra kontrol tekrar DO deyimine aktarılacaktır. i değişkeninin değeri m_3 kadar artırılacak ve tekrar n nolu deyime kadar olan işlemler işlenecektir. Kontrolun DO deyimine her dönüşünde i değişkeni m_3 (1) kadar artırılacak ve tekrarlama işlemi yapılacaktır. Bu işlemler i değişkeninin değeri m_2 (15) değerine eşit oluncaya kadar devam edilecektir. Yani NL=15 oluncaya kadar. i değişkeninin değeri m_2 değerinden büyük olunca kontrol n deyimini izleyen deyime aktarılacaktır.

Örneğin, aritmetik IF deyimi ile ilgili olarak önceki bölümde 1'den N'ye (N=10) kadar olan sayıların toplamını bulmak için;

```
      N=10
      K=1
      IT=0
15    IT=IT+K
      IF (K-N)  10, 20, 20
10    K=K+1
      GO TO 15
20    WRITE (6,25) IT, N
25    FORMAT (5X, I7, 5X, I3)
      STOP
      END
```

biçiminde yazılan program DO deyimi ile aşağıdaki biçimde yazılabilir.

Akış - Şeması




```
      N=10
      IT=0
      DO 15 K=1, N, 1
15    IT=IT+K
      WRITE (6,25) IT,N
25    FORMAT (5X, I7, 5X, I3)
      STOP
      END
```

Bu küçük programda DO döngüsü başladığı zaman K=1 olacak ve 15 nolu deyimin gerektirdiği toplama işlemi yapılacaktır. Yani, IT=0+1=1 olacaktır. Kontrol tekrar DO deyimine dönecek, K=2 olacak ve 15 nolu deyime göre IT=1+2=3 bulunacaktır. Tekrar DO deyimine dönülecek K=3 olacak ve yukarıdaki işlem biçimi K=10 olun-

caya kadar devam edecektir. K'nin bu değeri de toplama eklenecektir. Tekrar DO deyimine dönülürse ve K'nin değeri 1 artırıldığı zaman bu kez K=11 olacaktır. Fakat K değişkeninin alacağı en son değer m_2=N=10 olarak belirlendiği için tekrarlama olmayacak ve kontrol WRITE deyimine geçecektir. Bu deyime göre de hesaplanmış olan IT değişkeninin değeri N ile birlikte yazılacak ve programda sona erecektir.

2.12.1.1. DO Deyimi ve Döngü ile İlgili Kurallar

DO deyimi ve döngü konusunda uyulması gereken kurallar şöyle özetlenebilir:

1. DO deyimi genel kalıbına uygun olarak yazılmalıdır. Örneğin aşağıdaki deyimler geçerli olmayan, yanlış yazılmış deyimlerdir.

DO 20 J= −1, 40, 4	Başlangıç değer negatif olamaz.
DO 10 A=3, 28	Değişken gerçel olamaz.
DO 10 J(I) =1, 20,2	Değişken dizinli (indisli) olamaz.
DO 20, L=M,K,S	Deyim numarasından sonra virgül (,) olamaz. Artış parametresi (m_3) bir tamsayı değişkeni ya da sabiti olmalıdır. S gerçel değişkeni olamaz.

2. DO döngü alanının ilk ve son deyimleri işlenebilir bir deyim olmalıdır. Örneğin, FORMAT, DIMENSION ve tip bildirme deyimleri (INTEGER, REAL, DOUBLE PRECISION v.b.) olamaz.

3. DO döngüsünün son deyimi aşağıda belirtilen işlenebilir deyimlerden biri de olamaz.

. Tüm GO TO deyimleri
. Aritmetik IF deyimi
. DO deyimi
. STOP, PAUSE, RETURN deyimleri
. Yukarıdaki deyimlerden birini kapsayan mantıksal IF deyimleri de olamaz. Bunun dışındaki mantıksal IF deyimleri olabilir.

Eğer yukarıdaki deyimlerden birinin DO döngüsünün son deyimi olarak kullanılması zorunlu ise bu durumda döngünün son deyimi olarak CONTINUE deyimi kullanılır. CONTINUE deyimi kukla (dummy) bir deyim olup derlenmiş programda (amaç programda) her hangi bir işlem gerektirmez. Bu nedenle bir FORTRAN programında işlenebilir bir deyimin olduğu her yerde kullanılabilir. Bu özelliğinden dolayı çoğu kez DO döngüsünün son deyimi olarak CONTINUE deyimi kullanılır. Örneğin;

```
      :
      IT=0
      DO 10 I=1, 50
      IT=IT+I
   10 CONTINUE
      :
```

gibi. Böylece döngünün son deyimi ile ilgili kuralın sınırlılığı giderilmiş olur.

4. DO deyimi parametrelerinin (i,m_1,m_2,m_3) değerleri döngü içinde değiştirilemez ve yeniden belirlenemez. Örneğin;

```
      :
      DO 6 I=1,40
      :
      :
      I=20
    6 CONTINUE
      :
```

ifadesi geçerli değildir. Çünkü döngü içinde I=20 olarak tekrar belirlenmektedir. Ancak;

```
      :
      J=20
      DO 6 I=1,J
      :
      :
    6 CONTINUE
      :
```

ifadesi geçerlidir. Çünkü J'nin değeri işlemden önce bildirilmektedir.

5. Dışardan herhangi bir biçimde DO döngüsü içine kontrol aktarılmaz. Örneğin;

```
      :
   15 .....
      IF (S—A) 15,15,10
      DO 20 J=1,30
      :
   10 .....
   20 CONTINUE
      :
```

ifadesi geçerli değildir. Çünkü (S—A) ifadesinin sonucu pozitif olduğunda kontrol DO döngüsü içindeki 10 nolu deyime aktarılmaktadır.

6. Döngünün tekrarı tamamlanmadan döngüden çıkılabilir. Örneğin;

```
      :
      DO 20 I=2, 100,4
      :
      IF (N—I) 20, 15, 20
20    CONTINUE
15    .....
      :
```

ifadesi ile N=I (yani N—I=0) olunca döngüden çıkılmakta, aksi durumlarda 20 CONTINUE deyimine geçilip döngünün gerektirdiği sayıda tekrar yapılmaktadır. Döngü içinde aktarma deyimleri kullanıldığı zaman döngüden çıkmak istenilmeyen durumlarda, gerekli işlemlerden sonra kontrolun CONTINUE deyimine ulaştırılması sağlanmalıdır.

7. Birden fazla döngü iç içe yuvalanmış olarak konulabilir. Ancak böyle durumlarda içe yuvalanmış döngülerin deyimleri, dıştaki döngünün alanı içinde olmalıdır. Aşağıdaki örnekler geçerli olan döngülerdir. Örneklerde görüldüğü gibi birden fazla döngünün enson deyimleri aynı olabilir.

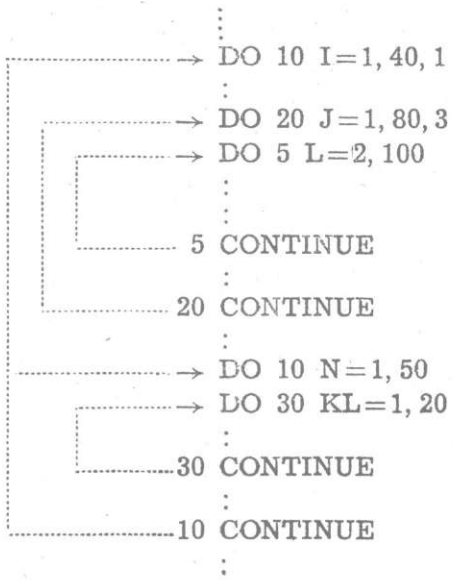
a)

```
      :
      DO 5 I=1, 100
      DO 10 J=1, 150
      :
      :
10    CONTINUE
      :
      :
5     CONTINUE
      :
```

b)

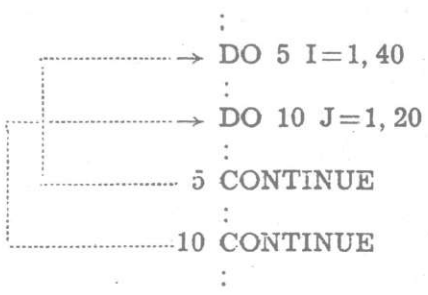
```
      :
      DO 20 I=1, 40
      :
      DO 20 K=1, 50
      :
      :
20    CONTINUE
      :
```

c)

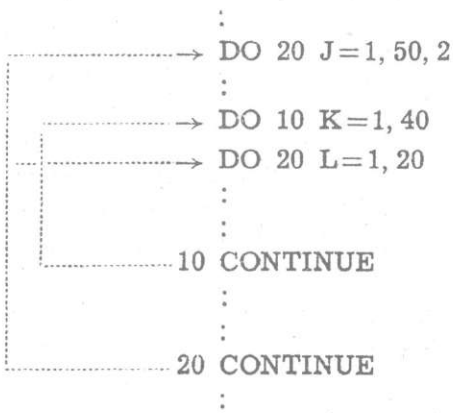


Aşağıdaki örnekler ise geçerli olmayan döngü örnekleridir :

a)



b)



Görüldüğü gibi iç içe yuvalanmış döngülerin alanı bir biri ile kesişmektedir. Yani uygun bir biçimde iç içe yuvalanmamışlardır.

8. İşlem kontrolü dıştaki döngüden içteki döngünün alanı içine aktarılmamalıdır. Fakat içteki döngüden dıştaki döngünün alanı içine kontrol aktarılabilir. Örneğin;

```
      :
      DO 20 I=1, 25
      IF (I—N) 5, 5, 10
5     .....
      DO 30 J=1, 28
      :
10    .....
      :
30    CONTINUE
      :
20    CONTINUE
      :
```

program bölümü hatalıdır. Çünkü (I—N) ifadesinin sonucu pozitif olduğu zaman kontrol içteki döngünün 10 nolu deyimine aktarılmaktadır.

Ancak aşağıdaki gibi yazılan bir program bölümü hatalı değildir.

```
      :
      DO 20 I=1, 25
      :
      DO 30 J=1, 50
      :
      IF (A.EQ.B) GO TO 10
      :
30    CONTINUE
      :
10    .....
      :
20    CONTINUE
      :
```

İç içe yuvalanmış döngülerde dıştaki döngünün bir kez tekrarında içteki döngü istenildiği kadar tekrarlanmaktadır. Yani dıştaki döngü bir tekrar yaparken içteki tüm tekrarlarını yapar. Örneğin;

```
      :
      L=0
      DO 1 I=1, 4
      DO 1 J=1, 5
      L=L+I+J
1     CONTINUE
      :
```

biçiminde içe içe yuvalanmış döngüler işlenirken dıştaki döngü bir kez tekrarlanırken içteki döngü 5 kez tekrarlanmaktadır. Böylece dıştaki döngü 4 kez tekrarlandığında içteki döngü de 20 kez tekrarlanmış olmaktadır. Yani L=L+I+J deyimide 20 kez tekrarlanmakta ve yuvalanmış döngüler terk edildiğinde L=110 olmaktadır. Bu program bölümünün işleyişi aşama aşama aşağıda verilmiştir.

Döngü Tekrar Sayısı	I	J	L+I+J=L	
1	1	1	0+1+1	2
2	1	2	2+1+2	5
3	1	3	5+1+3	9
4	1	4	9+1+4	14
5	1	5	14+1+5	20
6	2	1	20+2+1	23
7	2	2	23+2+2	27
8	2	3	27+2+3	32
9	2	4	32+2+4	38
10	2	5	38+2+5	45
11	3	1	45+3+1	49
12	3	2	49+3+2	54
13	3	3	54+3+3	60
14	3	4	60+3+4	67
15	3	5	67+3+5	75
16	4	1	75+4+1	80
17	4	2	80+4+3	86
18	4	3	86+4+3	93
19	4	4	93+4+4	101
20	4	5	101+4+5	110

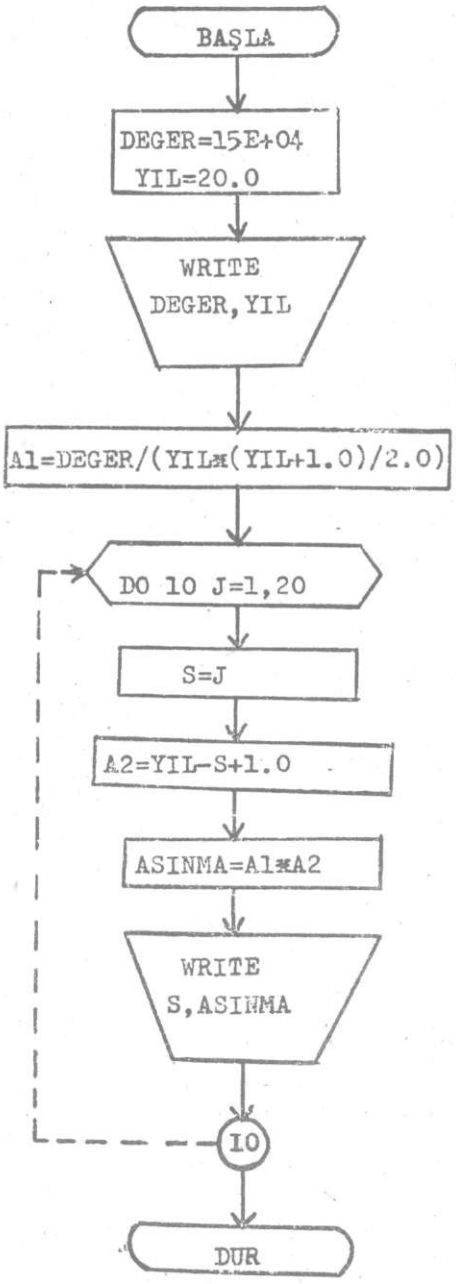
L=110

2.12.1.2 DO ile İlgili Program Örnekleri

Örnek 1 : Aşınmanın Hesaplanması

DO deyiminin sağladığı kolaylıkları görmek için aktarma deyimleri bölümünde verilen yıllık aşınmanın yıl basamakları toplamı yöntemi ile hesaplanması örneği burada bir de DO deyimi ile yapılacaktır. Ancak bu kez sermaye malının ömrünü YIL değişkeni ile göstere-

lim ve YIL=20 olduğunu varsayalım. DEGER ve ASINMA değişkenleri önceden olduğu gibi sırası ile sermaye malının değerini ve yıllık aşınmaları göstersin. Buna göre hazırlanmış akış-şeması ve program aşağıda verilmiştir.



Program :

```
C  ASINMANIN HESAPLANMASI
   15 FORMAT (1H1,/ /, 5X, 'SERMAYE DEGERI=',
      1F9.0, 'TL.', / /, 5X, 'SERMAYENIN OMRU=',
      1F4.0, 'YIL', / /, 5X, 'YIL', 5X, 'ASINMA PAYI')
   20 FORMAT (5X, F4.0, 5X, F11.2)
      DEGER=15E+04
      YIL=20.0
      WRITE (6,15) DEGER, YIL
      A1=DEGER/(YIL★ (YIL+1.0)/2.0)
C  DONGUDE S'YE J DEGERI VERMEK VE
C  ASINMAYI HESAPLAMAK
      DO 10 J=1,20
      S=J
      A2=YIL—S+1.0
      ASINMA=A1★A2
      WRITE (6,20) S, ASINMA
   10 CONTINUE
      STOP
      END
```

Bu programda yıl sayıları J değişkeni ile saydırılmakta ve S değişkenine J değerini atamakla (yıl sayıları gerçel biçime dönüştürülerek) A2 değeri hesaplanmaktadır. Ayrıca görüldüğü gibi WRITE deyimlerinin FORMAT deyimleri programın başına konulmuştur. Bu tür uygulama program işleyişini değiştirmez. Çünkü FORMAT deyimleri işlenemez deyim olup programın istenilen kısmına konulabilir.

Örnek 2 : Sıra Bekleme Sisteminin Değerlendirilmesi
(Kuyruk Teorisi)

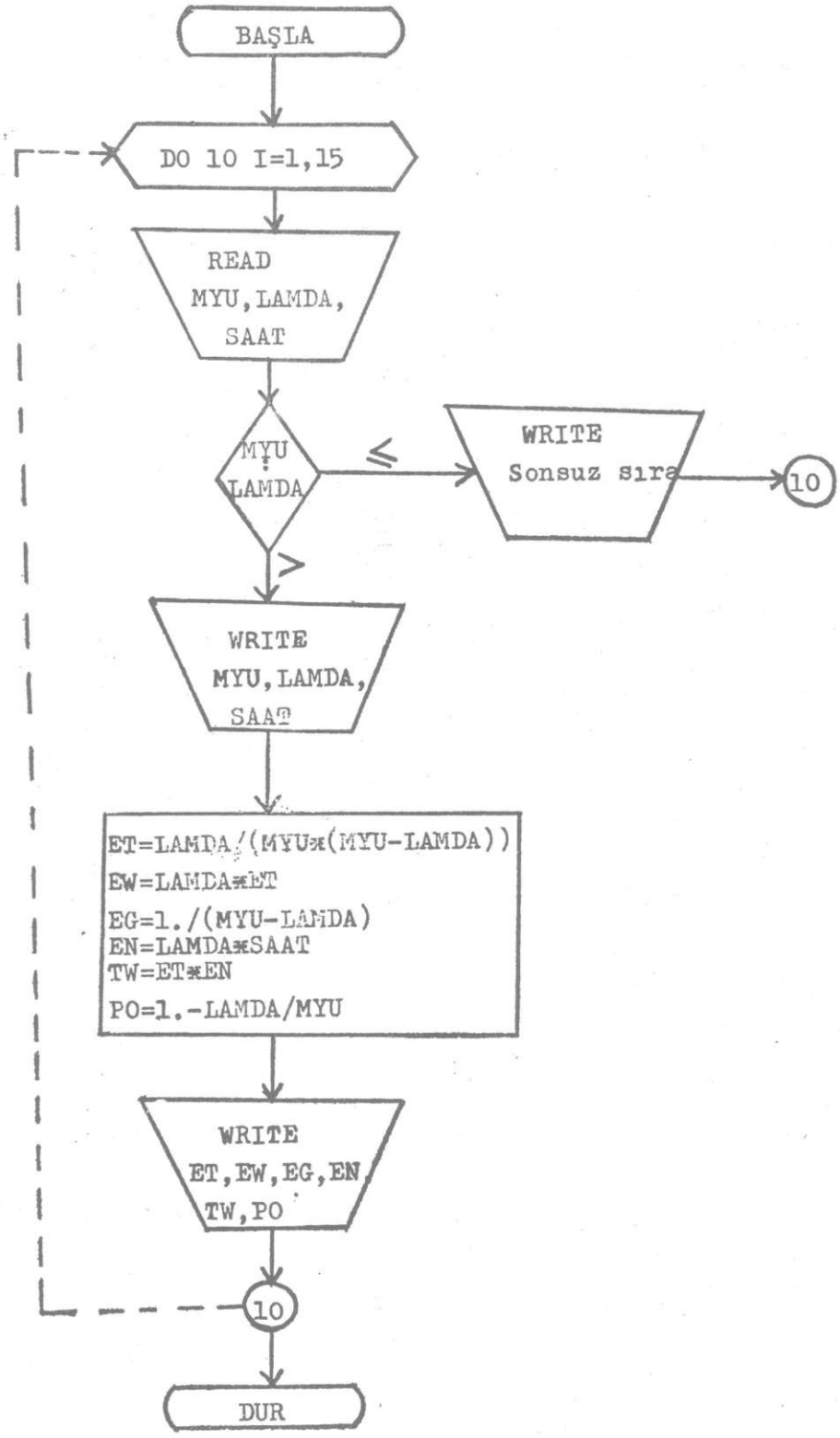
Tek kanallı ve tek aşamalı bir sıra bekleme sistemi olan bir işletmede (yani işletmeden hizmet isteminde bulunan müşterilerin bir tek sıra oluşturduğu ve hizmetin de bir tek servis istasyonundan sunulduğu bir işletmede) ortalama olarak bir saatte hizmet isteminde bulunan müşteri sayısı λ (LAMDA) ve himet edilen müşteri sayısıda m (MYU) ise, LAMDA ve MYU'nun Poisson bir dağılım biçimi gösterdiği varsayımı ile, bu işletmede belli bir dönemde, örneğin 8 saatlik bir iş gününde (SAAT) ortalama olarak :

a) Gelen müşternin sırada bekleyeceği zaman (ET),
b) Sırada bekleyecek müşteri sayısı (EW),
c) Sistemde geçecek zaman (EG), (yani ET+hizmet süresi),
d) Toplam gelen müşteri sayısı (EN),
e) Toplam sırada bekleme zamanı (TW),
f) Sistemin boş kalma olasılığı (PO) nedir?

Tek kanallı belirsiz evrenli ve Poisson dağılımlı bir sıra bekleme modelinde hesaplamalar aşağıdaki yöntemi izler.

ET=LAMDA/(MYU★(MYU−LAMDA))
EW=LAMDA★ET
EG=1/(MYU−LAMDA)
EN=LAMDA★SAAT
TW=ET★EN
PO=1−LAMDA/MYU

Bu modele göre hesapların yapılabilmesi için modelin temel varsayımı olan bir koşulun karşılanması gerekir. Bu koşulda, MYU değerinin LAMDA değerinden büyük olmasıdır. Aksi durumda hizmet için sıra bekleyen müşteri sayısı kuramsal olarak sonsuza doğru büyür. Bu nedenle sıra bekleme sisteminin değerlendirilmesi için MYU'nun LAMDA'dan büyük olması gerekir. Bu koşul sağlanmazsa değerlendirme işlemine belirtilen model uygulanamaz. Başka bir yaklaşım gerektirir. Bu açıklamalardan sonra 15 veri kümesi için yani 15 farklı sıra bekleme sisteminin verilerini değerlendirecek bir program bitişik sayfada verilen akış şemasına dayalı olarak yazılabilir. Akış şemasında görüldüğü gibi her veri kümesi için tüm işlemler DO döngüsü içinde yapılmaktadır. Döngü 15 kez tekrarlandığında tüm sıra bekleme sistemleri için gerekli işlemler yapılmaktadır. MYU'nın LAMDA'ya eşit ya da daha büyük olduğu sistemler için her hangi bir hesaplama yapılmamaktadır.


BAŞLA
DO 10 I=1,15
READ
MYU, LAMDA,
SAAT
MYU
LAMDA
≤
WRITE
Sonsuz sıra
10
>
WRITE
MYU, LAMDA,
SAAT
ET=LAMDA/(MYU*(MYU-LAMDA))
EW=LAMDA*ET
EG=1./(MYU-LAMDA)
EN=LAMDA*SAAT
TW=ET*EN
PO=1.-LAMDA/MYU
WRITE
ET, EW, EG, EN,
TW, PO
10
DUR

```
C  SIRA BEKLEME SISTEMINI DEGERLENDIRME
C
      REAL MYU, LAMDA
   20 FORMAT (3F7.2)
   30 FORMAT (/ /, 15X, 'SONSUZ SIRA')
   40 FORMAT (/,6X, 'LAMDA', 5X, 'MYU', 8X, 'SAAT')
   50 FORMAT (/,1X, 3F10.2)
   60 FORMAT (/,7X, 'ET', 6X, 'EG', 6X, 'EN', 6X,
     1'TW', 6X, 'EW', 6X, 'PO')
   70 FORMAT (/, 1X, 6F8.2)
      DO 10 I=1, 15
      READ (5, 20) MYU, LAMDA, SAAT
      IF (MYU-LAMDA) 75, 75, 80
   75 WRITE (6,30)
      GO TO 10
   80 WRITE (6,40)
      WRITE (6,50) MYU, LAMDA, SAAT
      ET=LAMDA/(MYU*(MYU-LAMDA) )
      EW=LAMDA*ET
      EG=1.0/(MYU-LAMDA)
      EN=LAMDA*SAAT
      TW=ET*EN
      PO=1.0-LAMDA/MYU
      WRITE (6,60)
      WRITE (6,70) ET, EG, EN, TW, EW, PO
   10 CONTINUE
      STOP
      END
```

2.12.2 Dizi ve Dizinli Değişken Kavramları

Dizi, sıralı birimlerden ya da elemanlardan oluşan bir kümedir. Dizinin elemanları aynı adı taşımakla birlikte dizi sırası içindeki yerlerini belirtmek için her eleman bir sıra numarası taşır. Sıra numarası elemanın altında belirtilir. Buna dizin adı verilir. Böyle yazılmış olan değişkenlerede dizinli değişkenler denilir. Dizinli değişkenler gerçel ya da tamsayılı değişkenler olabilirler.

Örneğin, bir işletmenin belli bir dönemdeki satış kalemlerini belirten aşağıdaki verileri düşünelim.

Satışlar :
2000
3600
1700
6450
5000

Yukarıdaki dizi satış kalemlerini belirten elemanlardan oluşmuştur. Satışlar S_i dizinli değişkeni ile gösterilirse S_3 değişkeni üçüncü satış kalemini, yani 1700 değerini belirtir. Aynı şekilde;

S_i	$(i=1,2,\ldots\ldots,5)$
$S_1=2000$	
$S_2=3600$	
$S_3=1700$	
$S_4=6450$	
$S_5=5000$	

değerlerini belirtirler.

Diziler yukarıda verilen örnekte görüldüğü gibi tek boyutlu ya da çok boyutlu olabilirler. Örneğin, bir işletmenin satışlarını belirten aşağıdaki tabloda satırlar yılları, sütunlarda bölgeleri göstersin.

SATIŞLAR

Bölgeler :

Yıllar	1	2	3	4
1978	1650	2000	4000	8600
1979	3200	4340	6250	1500
1980	7600	8420	9360	675

Satırları (yani yılları) i ile sütunları da (bölgeleri) j ile gösterirsek, satışları belirtecek S değişkeni $S_{i,j}$ biçiminde yazılabilir. Bu örnekte $i=1, 2, 3$ ve $j=1, 2, 3, 4$ değerlerini almaktadırlar. Buna göre :

$S_{1,1}=1650$	$S_{1,2}=2000$	$S_{1,3}=4000$	$S_{1,4}=8600$
$S_{2,1}=3200$	.....	.....	$S_{2,4}=1500$
$S_{3,1}=7600$	.....	.....	$S_{3,4}=675$

değerlerini belirtirler. Başka bir deyişle $S_{i,j}$ bir matris olup

$$S_{i,j}=\begin{bmatrix} 1650 & 2000 & 4000 & 8600 \\ 3200 & 4340 & 6250 & 1500 \\ 7600 & 8420 & 9360 & 675 \end{bmatrix} = \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} & S_{1,4} \\ S_{2,1} & S_{2,2} & S_{2,3} & S_{24} \\ S_{3,1} & S_{3,2} & S_{3,3} & S_{3,4} \end{bmatrix}$$

biçimlerinde de gösterilebilir. $S_{i,j}$ matrisi $(3\times4=12)$ 12 elemanlı ve iki boyutlu bir matris olup i matrisin sırasını j' de sütunlarını belirtmektedir. Buna göre $S_{2,3}$ dizinli değişkeni $S_{i,j}$ matrisinin ikinci sırasının üçüncü elemanını (6250) belirtir. Dizinli değişkenler FORTRAN dilinde dizinler ayraç içine alınarak yazılır. Örneğin;

S_1 değişkeni S(I) biçiminde,

$S_{i,j}$ değişkeni S(I, J) biçiminde,

$S_{2,3}$ değişkeni S(2,3) biçiminde yazılır.

2.12.2.1 Dizinlerle İlgili Kurallar

Dizinli değişken oluştururken aşağıdaki kuralların göz önüne alınması gereklidir.

1. Dizinler tamsayılı sabitler, değişkenler ya da aritmetik ifadeler olmalıdır. Ancak dizinler gerçel olduğu durumlarda tamsayılı biçime dönüştürülürler. Bu kurala göre geçerli olan bazı dizin biçimleri aşağıda verilmiştir.

a) Bir sabit : S(1)
K(3)

b) Bir değişken : A(I)
J(L)
B(KAR)

c) Bir değişken ∓ bir sabit : S(I+4)
K(L+2)
A(J−1)
X(NYIL+3)

d) Bir sabit ★ bir değişken : B(3★I)
K(2★J)
K(2★LIR)

e) Bir sabit ★ bir değişken ∓ bir sabit. VR(2★K+8)
J(4★L+10)
S(5★J−6)
I(2★M−4)
B(2★KH+3)

Yukarıdaki c,d,e maddelerinde verilen örneklerde ayraç içindeki aritmetik ifadelerin sonucu dizinin değerini oluşturur. Örneğin, I=1 ise S(I+4) değişkeni S(5) olarak göz önüne alınır. Yukarıdaki kurallara göre aşağıdaki dizinli değişkenler geçerli değildir :

S(4+I)
SM(1−J)
B(I★3)

2. Bir dizinin değeri negatif ya da sıfır olamaz. Yani A(0), A(−K), B(−2), K(0) v.b. ifadeler geçerli değildir.

3. Bir değişkenin dizin sayısı değişkenin kaç boyutlu olduğunu gösterir. Dizinli bir değişken en az bir en çok 7 boyutlu olabilir. Birden fazla olan dizinler virgül (,) ile bir birinden ayrılırlar. Buna göre: S(I) ve B(3) tek boyutlu, A(I,J) ve K(M,N) iki boyutlu, A(I,J,K) üç ve R(I,L,M,J,K) beş boyutlu dizinli değişkenlerdir. S(I)(J) ve A(2,3, 5,6,18,7,1,4) geçerli olmayan dizinli değişkenlerdir. Çünkü birinci değişkende virgül kullanılmamış ve ayraçlar da hatalı yazılmıştır. İkinci değişkeninde dizin sayısı 8'dir. Yani 7'den fazladır.

4. Dizinli bir değişken bir başka değişken için dizin olabilir. Örneğin, S(I,J(K,L)) değişkeninde, J(K,L) değişkeni aslında kendisi bir dizinli değişkendir ve S değişkeninin ikinci dizini olmaktadır. Başka bir ifade ile J(K,L) değişkeninin değeri S değişkeninin ikinci dizini olmaktadır.

2.12.2.2. DIMENSION Deyimi ve Dizinli Değişkenlerde Girdi/Çıktı

Dizinli değişkenler program içinde her türlü bilgisayar işlemine konulabilir. Çoğu zaman bir dizinli değişkenin kullanımı büyük kolaylık sağlar. Ancak programda dizinli değişkenlerin kullanıldığı durumlarda, hangi değişkenin ya da değişkenlerin dizinli olduğunu ve dizinli değişkenin kaç boyutlu ve elemanlı olduğunun programın başlangıcında belirtilmesi gerekir. Böylece bilgisayar dizinli değişkenin elemanları için gerekli sayıda bellekte yer ayırır. Bunu sağlamak için de DIMENSION deyimi kullanılır.

DIMENSION deyimi derleyiciye bilgi veren işlenemez bir deyim olup genel kalıbı;

DIMENSION $d_1,d_2,d_3,\ldots\ldots$

biçimindedir. Sembolik d'ler ayraç içinde boyutları ve eleman sayıları belirtilen virgül ile bir birinden ayrılmış dizinli değişken listesini temsil etmektedirler. Örneğin;

DIMENSION S(20), A(40), X(10, 20)

deyimi ile derleyiciye S ve A değişkenlerinin tek boyutlu sırası ile 20 ve 40 elemanlı dizinli değişkenler; X değişkeninin de iki boyutlu 200 elemanlı ($10\times20=200$) bir dizinli değişken olduğu bildirilmektedir.

Dizinli değişkenlerin program içinde kullanımı dizinli olmayan değişkenler gibi olmasına karşın bunların bilgisayara aktarılması yani okutulması ve depolanmış değerlerinin yazdırılması biraz farklılık gösterir. Okutma ve yazdırma işlemleri için çeşitli yöntemler kullanılır. Ancak burada dizinli değişkenlerin okutulması ve yazdırılması için en çok kullanılan beş yöntem ele alınacaktır.

1. Normal DO Deyimi Aralıcılığı ile Girdi/Çıktı :

Örneğin, 30 elemanı olan dizinli bir S(I) değişkeni DO deyimi aracılığı ile okutmak istenilirse aşağıdaki ifade yazılabilir.

```
      DIMENSION S(30)
      DO 3 I=1,30
    3 READ (5,10) S(I)
   10 FORMAT (F8.2)
      ⋮
```

Yukarıdaki ifadede DIMENSION deyimi ile S(I) dizinli değişkeninin 30 elemanlı olduğu bildirilmekte ve DO döngüsü içine konularak okunması sağlanmaktadır. DO döngüsü 30 kez tekrarlanırken READ deyimi de 30 kez tekrarlanmakta ve her seferinde S(I) değişkeninin elemanları S(1), S(2), S(3),..... S(30) biçiminde tek tek okunmaktadır. Örnekte görüldüğü gibi saydırma işlemini yapan I okutulan S dizisinin dizini olarak belirtilmiştir. Yukarıdaki okutma işlemi için 30 veri kartı kullanılacaktır. Yani her kart bir elemanın sayısal değerini ilk sekiz sütununda taşıyacaktır.

Eğer yukarıda verilen FORMAT deyimi;

10 FORMAT (10F8.2)

biçiminde yazılmış olsaydı, okutma işlemi yine 30 kartta yapılacaktır. Çünkü her döngü tekrarında READ deyimi ile karşılaşılınca S(I) dizisinin bir elemanı veri kartının ilk 8 sütunundan okunacaktır.

Eğer S(I) dizisinin değerleri yazdırılmak istenirse aynı kalıp kullanılır.

```
      DIMENSION S(30)
      DO 5 I=1,30
    5 WRITE (6,20) S(I)
   20 FORMAT (3X, F10.2)
      :
```

ifadesi ile S(I) dizisinin 30 elemanı alt alta 30 satıra yazılır.

S(I) dizisinin okutulması ve yazdırılması aynı programda yapılıyorsa DIMENSION S(30) deyiminin program başında bir kez yazılması yeterlidir. Yani;

```
      DIMENSION S(30)
      DO 3 I=1,30
    3 READ (5,10) S(I)
   10 FORMAT (F8.2)
      :
      :
      DO 5 I=1,30
    5 WRITE (6,20) S(I)
   20 FORMAT (3X, F10.2)
      :
```

biçiminde yazılabilir. Eğer gerekli düzenlemeler yapılırsa tek FORMAT deyimi de kullanılabilir. Örneğin, READ deyimi 20 nolu FORMAT için 3 READ (5,20) S(I) olarak yazılırsa 10 nolu FORMAT'a gerek yoktur. Yani hem okutma hem de yazma işlemi 20 FORMAT ile yapılabilir. Kuşkusuz verilerin kartlara delinmesi de bu FORMAT'a göre yapılacaktır. Yani her kartın ilk üç sütunu boş bırakılacak ve her elemanın değeri 10 sütuna delinecektir.

2. Kapalı DO Döngüsü ile Girdi/Çıktı :

Biraz önce DO döngüsü ile yapılan okutma işlemi aynı biçimde;

```
      DIMENSION S(30)
      READ (5,10) (S(I), I=1,30)
   10 FORMAT (10F8.2)
      :
```

ifadesi ile üç kartta yapılabilir. Burada kapalı DO döngüsü,

```
READ (5,10) (S(I), I=1,30
```

ifadesidir. Başka bir örnek olarak;

```
      DIMENSION X(10)
      WRITE (6,7) (X(J), J=1,10)
    7 FORMAT (10F12.2)
      :
```

ifadesi ile X(J) dizisinin 10 elemanı tek satıra arka arkaya yazılacaktır. Yani satır atlamadan F12.2 belirleyicisi 10 kez tekrarlanacaktır. Bu da 10 elemanlı X(J) dizinli değişkenini yazdırmak için yeterli olmaktadır.

3. Çift Normal ya da Kapalı DO Döngüleri ile Girdi/Çıktı :

A(I,J) dizinli değişkeni ile belirtilen 10 sıralı (I=10) ve 20 sütunlu (J=20) bir veri tablosunu okutmak isteyelim.

Normal DO döngüsünü kullanırsak;

```
      DIMENSION A(10,20)
      DO 5 I=1,10
      DO 5 J=1,20
    5 READ (5,10) A(I,J)
   10 FORMAT (F4.0)
      :
```

deyimleri ile A tablosu ya da matrisi aşağıdaki gibi okunur :

(1-20)	kartlardan	A(1,1)	A(1,2)	A(1,3)	.....A(1,20)
(21-40)	»	A(2,1)	A(2,2)	A(2,3)	.....A(2,20)
(41-60)	»	A(3,1)	A(3,2)	A(3,3)	.....A(3,20)
:		:	:	:	:
:		:	:	:	:
(181-200)	»	A(10,1)	A(10,2)	A(10,3)	.....A(10,20)

Örnekte görüldüğü gibi dış döngü bir kez tekrarlanırken içteki döngü 20 kez tekrarlanmaktadır. Yani dış döngüde I'nın değeri her seferinde 1 artarken iç döngüdeki J'nin değeri 1'den 20'ye kadar gitmektedir. Böylece önce birinci sıra, sonra ikinci sıra v.b. şeklinde okunmaktadır. Son aşamada I=10 olduğu zaman onuncu sıra okunacak ve iç döngü 200 kez (10×20=200) tekrarlanmış olacaktır.

10 FORMAT (F4.0) deyiminde görüldüğü gibi 200 okutma işlemi yani 200 elemanın okutulması 200 karttan yapılmaktadır. Böylece tüm okutma işlemi için 200 kart kullanılmaktadır. Ayrıca okutma ifadelerine göre önce her sıranın sütunları sıra ile okunduğu için veri tablosunun her sırasının her elemanı bir kartta delinmelidir. Yani 1 sıra için 20 kart delinecektir.

Yukarıda verilen okutma işlemi aynı biçimde çift kapalı DO deyimleri ile de aşağıdaki gibi yazılabilir.

```
      DIMENSION A(10,20)
      READ (5,10) ( (A(I,J), J=1,20), I=1,10)
   10 FORMAT (20F4.0)
      :
```

Burada J=1,20 iç döngüsü, I=1,10'da dış döngüsünü belirtmektedirler. Okutma işlemi bu kez her sıra bir kartta olmak üzere 10 kartta yapılacaktır. Yani okutma işlemi aşağıdaki gibi olacaktır.

```
1. Karttan   A( 1,1)   A( 1,2)   A( 1,3)   .....A( 1,20)
2. Karttan   A( 2,1)   A( 2,2)   A( 2,3)   .....A( 2,20)
3. Karttan   A( 3,1)   A( 3,2)   A( 3,3)   .....A( 3,20)
:            :         :         :            :
:            :         :         :            :
:            :         :         :            :
:            :         :         :            :
10. Karttan  A(10,1)   A(10,2)   A(10,3)   .....A(10,20)
```

Ancak okutma ifadesi,

```
      DIMENSION A(10,20)
      READ (5,10) ( (A(I,J), I=1,10), J=1,20)
   10 FORMAT (10F4.0)
```

biçiminde yazılırsa bu kez 20 veri kartı kullanılacaktır. Fakat bu sefer sütunlar sıra ile okunacaktır. Yani 10 FORMAT (10F4.0) ifadesine göre her sütunun elemanları bir tek karttan aşağıdaki gibi okunacaktır.

```
1. karttan   A( 1,1)   A( 2,1)   A( 3,1)   .....A(10,1)
2. karttan   A( 1,2)   A( 2,2)   A( 3,2)   .....A(10,2)
3. karttan   A( 1,3)   A( 2,3)   A( 3,3)   .....A(10,3)
:            :         :         :            :
:            :         :         :            :
:            :         :         :            :
20. karttan  A(1,20)   A(2,20)   A(3,20)   .....A(10,20)
```

4. Normal ve Kapalı DO Döngüleri Birlikte Girdi/Çıktı :

Örneğin, B(15,10) dizinli değişkeni ile gösterilen iki boyutlu, 150 elemanlı ($15 \times 10 = 150$) ve bilgisayara okutulmuş bir tablonun değerleri yazdırılmak istensin. Normal ve kapalı DO döngüleri birlikte kullanılarak yazdırma işlemi için aşağıdaki ifadeler yazılabilir.

```
      DIMENSION B(15,10)
      DO 5 I=1,15
    5 WRITE (6,20) (B(I,J), J=1,10)
   20 FORMAT (10F9.2)
      :
```

Burada J=1,10 iç döngü olup tablonun 10 sütunlu bir sırasını bir satıra yazar. Böylece veriler 15 satırda alt alta yazılır.

2.12.2.3 Dizinli Değişkenlerle İlgili Program Örnekleri

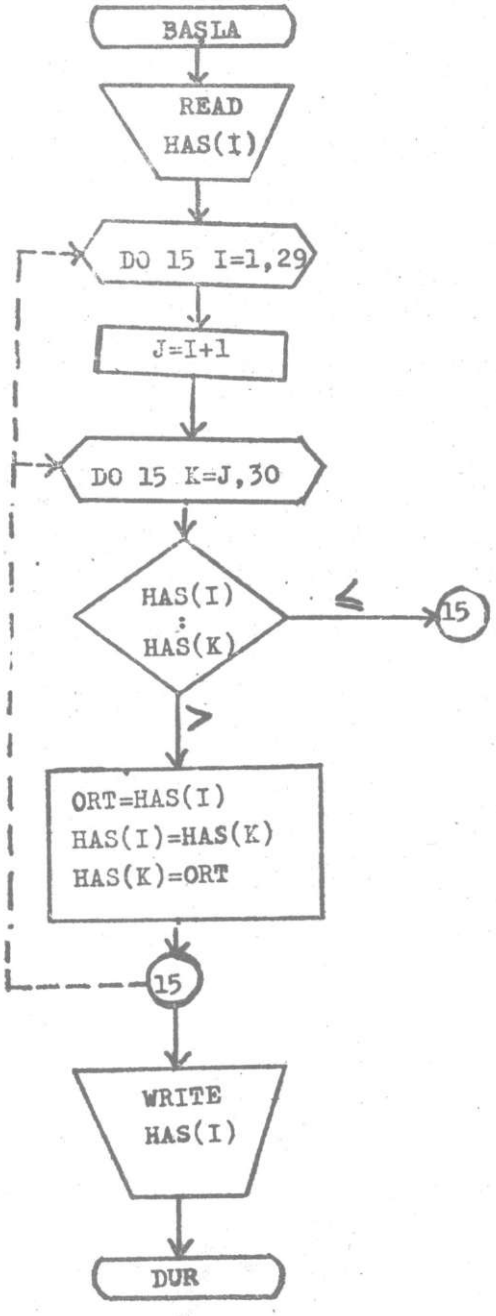
Örnek 1 : Bir dizideki sayıların küçükten büyüğe doğru sıralanması

Bir işletmenin bir aydaki günlük satış hasılatlarını küçükten büyüğe doğru sıralayan bir program hazırlayalım. Günlük satış hasılatlarını 30 elemanlı HAS(I) dizinli değişkeni ile gösterelim.

İzlenen Yöntem : Önce HAS (1), HAS (2),....., HAS (30) elemanlarının en küçüğünü buluyoruz. En küçük değere HAS (K) diyelim. HAS (1)'in değerini HAS (K)'ye HAS (1)'e de HAS (K) değerini verelim. Yani yer değiştiriyoruz. Böylece HAS (1) en küçük değerli eleman oluyor. Sonra HAS (2), HAS (3),....., HAS (30) elemanlarının en küçüğünü buluyoruz ve yine yer değiştirip bulunan en küçük değeri HAS (2)'ye veriyoruz. En son aşamada HAS (29) ile HAS (30) elemanlarının en küçüğünü buluyoruz ve tekrar yer değiştirip HAS (29)'a en küçük değeri HAS (30)'a da diğer büyük değeri veriyoruz. Böylece HAS (1) en küçük, HAS (2) ikinci küçük,....., HAS (30)'da en büyük değerleri almış olurlar. Yer değiştirme işlemi için ortak bir değişken olan ORT kullanılacaktır.

Her aşamada iki eleman karşılaştırıldığından dizinin sıra sayımı için iki dizin değişkeni I ve K kullanılacaktır. Birinci dizin değişkeninin sayma için alacağı en son değer dizideki eleman sayısının bir eksiği olacaktır, yani I=1,2,....,29 olacaktır. Dizinin son elemanını 2 ya da I+1 değeri ile başlatılacak olan K değişkeni sayacaktır. Yani K=2,3,....,30 biçiminde sayacaktır. Böylece başlangıçta I=1 ve K=2 iken en son aşamada I=29 ve K=30 olacaktır ve tüm elemanlar ikişer ikişer karşılaştırılmış olacaktır.

Bu açıklamalara göre hazırlanmış olan akış-şeması ve akış-şemasına dayalı olarak yazılmış olan program bitişik sayfada verilmiştir. Program işleyişini açıklığa kavuşturmak için örnek olarak beş elemandan oluşan bir dizi, program akışına göre işlenecek ve her aşamada değişkenlerin aldıkları değerler bir tabloda gösterilecektir.



Program :

```
C  SIRALAMA PROGRAMI
       DIMENSION HAS(30)
       READ (5,10) (HAS (I),I=1,30)
   10  FORMAT (10F8.2)
       DO 15 I=1,29
       J=I+1
       DO 15 K=J,30
       IF (HAS (I)—HAS (K) )15,15,20
   20  ORT=HAS (I)
       HAS (I) =HAS (K)
       HAS (K) =ORT
   15  CONTINUE
       WRITE (6,25) (HAS (I), I=1,30)
   25  FORMAT (5X, F10.2)
       STOP
       END
```

Programın akışını açıklamak için HAS (I) dizisinin beş elemanlı ve aşağıdaki değerleri olduğunu varsayalım.

HAS(1) =13. HAS(2)=19. HAS(3)=12. HAS(4)=10. HAS(5)=18.

Bu verilere göre birinci normal döngüde (I=1,4) ikinci normal döngüde döngü alanı (K=J,5) olacaktır. Bu döngülerin işleyişi ve değişkenlerin değerleri bitişik tabloda verilmiştir. Bu örnekte belirtilmek istenen nokta bir dizinin elemanlarının iki farklı dizin (I ve K) kullanılarak nasıl karşılaştırılabileceğidir. Programın çıktısı olarak HAS(I) dizisinin elemanları küçükten büyüğe doğru sıralanmış olarak alt alta 30 satırda verilecektir.

Sıralama Programının Sayısal İşleyişi

I	K = 2	K = 3	K = 4	K = 5
1	HAS (1)=13 HAS (2)=19 HAS (3)=12 HAS (4)=10 HAS (5)=18 I=1 J=2 K=2 IF(13-19)15,15,20 20 ORT=13 HAS (1)=12 HAS (3)=13 15 CONTINUE	HAS (1)=13 HAS (2)=19 HAS (3)=12 HAS (4)=10 HAS (5)=18 I=1 J=2 K=3 IF(13-12)15,15,20 20 ORT=13 HAS (1)=12 HAS (4)=13 15 CONTINUE	HAS (1)=12 HAS (2)=19 HAS (3)=13 HAS (4)=10 HAS (5)=18 I=1 J=2 K=4 IF(12-10)15,15,20 20 ORT=12 HAS (1)=10 HAS (4)=12 15 CONTINUE	HAS (1)=10 HAS (2)=19 HAS (3)=13 HAS (4)=12 HAS (5)=18 I=1 J=2 K=5 IF(10-18)15,15,20 15 CONTINUE
2	HAS (1)=10 HAS (2)=19 HAS (3)=13 HAS (4)=12 HAS (5)=18 I=2 J=3 K=3 IF(19-13)15,15,20 20 ORT=19 HAS (2)=13 HAS (3)=19 15 CONTINUE	HAS (1)=10 HAS (2)=13 HAS (3)=19 HAS (4)=12 HAS (5)=18 I=2 J=3 K=4 IF(13-12)15,15,20 ORT=13 HAS (2)=12 HAS (4)=13 15 CONTINUE	HAS (1)=10 HAS (2)=12 HAS (3)=19 HAS (4)=13 HAS (5)=18 I=2 J=3 K=5 IF(12-18)15,15,20 15 CONTINUE	
3	HAS (1)=10 HAS (2)=12 HAS (3)=19 HAS (4)=13 HAS (5)=18 I=3 J=4 K=4 IF(19-13)15,15,20 20 ORT=19 HAS (3)=13 HAS (4)=19 15 CONTINUE	HAS (1)=10 HAS (2)=12 HAS (3)=13 HAS (4)=19 HAS (5)=18 I=3 J=4 K=5 IF(13-18)15,15,20 15 CONTINUE		
4	HAS (1)=10 HAS (2)=12 HAS (3)=13 HAS (4)=19 HAS (5)=18 I=4 J=5 K=5 IF(19-18)15,15,20 20 ORT=19 HAS (4)=18 HAS (5)=19 15 CONTINUE			

SONUÇ :
HAS (1)=10.
HAS (2)=12.
HAS (3)=13.
HAS (4)=18.
HAS (5)=19. **olarak bulunur.**

Örnek 2 : İç Kârlılık Oranının Hesaplanması

İç kârlılık oranı bir yatırımın ömrü boyunca sağlayacağı gelirlerin bugünkü değeri ile maliyetin bugünkü değerini birbirine eşit kılan iskonta oranıdır. Yatırımın aynı yıl yapıldığını varsayarsak, kârlılık oranı aşağıdaki formül aracılığı ile bulunur.

$$YBD = \frac{G_1}{(1+r)} + \frac{G_2}{(1+r)^2} + \ldots\ldots + \frac{G_n}{(1+r)^n} + \frac{A}{(1+r)^n}$$

Burada;

YBD : Yatırımın bugünkü değerini,

G : Yatırımın yararlı ömrü boyunca sağladığı yıllık gelirler,

n : Yatırımın yararlı ömrü,

A : Yatırımın yararlı ömrü sonundaki artık (hurda) değeri,

r : Hesaplanacak olan iç kârlılık oranını belirtmektedirler. Yukarıdaki formül,

$$\left(\sum_{n=1}^{k} \frac{G_n}{(1+r)^n}\right) + \frac{A}{(1+r)^k} - YBD = 0 \qquad (n=1,2\ldots,k)$$

biçiminde yazılabilir.

Bu formül aracılığı ile (r)'yi bulmak için deneme yanılma yolu izlenir. Şöyleki; önce küçük bir (r) alınır, örneğin 0.01 gibi. Bu oran formülde yerine konulur ve eşitliğin sol tarafındaki değer hesaplanır. İlk aşamada (r) çok küçük seçildiği için bu değer pozitif olacaktır. Bu sonuç, yatırımın ömrü boyunca sağladığı gelirler ile hurdanın bugünkü değerlerinin yatırımın bugünkü değerinden büyük olduğunu ifade eder. Sonucu sıfır yapmak için (r) değeri artırılır. Aynı işlemler tekrarlanır ve eşitliğin sol tarafı sıfır ya da negatif değer alıncaya kadar bu tekrarlama devam eder. Negatif değer bulunduğu zaman (r)'nin değeri küçültülerek sonucu sıfır yapan (r) değeri aranır. Böylece belli bir tekrar sayısından sonra yukarıdaki eşitliği sıfır yapan (r) değeri (yaklaşık olarak da olsa) bulunacaktır. Kuşkusuz, tekrar sayısını azaltmak için çeşitli yöntemler kullanılabilir (örneğin, enterpolasyon).

Ancak bilgisayar tekrarlamalı işlemlerde son derece etkin olduğu için programlama amacı ile yukarıda belirtilen deneme yanılma yolu, her aşamada (r) değeri 0.01 oranında artırılarak izlenecektir. Böylece sıfır değeri bulunduğu zaman arama işlemi duracaktır. Eğer negatif değer bulunursa, (r) değeri yaklaşık olarak bir önceki değeridir. Yani, r=r—0.01 olarak belirlenecektir. İstenilirse enterpolasyon yöntemiyle kesirli (r) değeride bulunabilir, ya da r oranı 0.001 artırılarak aranır.

Çözüm için izlenecek formülü oluşturan ifadeler aşağıda yazıldığı gibi düşünülürse;

$$GBD = \sum_{n=1}^{k} \frac{G_n}{(1+r)^n}$$

$$HBD = \frac{A}{(1+r)^k}$$

$$TBD = GBD + HBD$$

programda kullanılacak değişkenler şöyle belirtilebilir :

YBD : Daha önce olduğu gibi, yatırımın bugünkü değeri.

G(N) : Yatırım sağladığı yıllık gelirler. N=1,2,...,K

L : İç kârlılık oranının dağılma alanı. Örneğin, oran %1 ile % 100 arasında aranabilir.

J : L alanı içerisinde arama yapılırken R oranının alacağı artış değeri. Örneğin eğer R iç kârlılık oranı % 1 ile % 100 arasında % 1 artırılarak aranılıyorsa, J=1,2,3.....100'dür. Ya da J=1,2,..., L.

S : J değerini gerçel yapan değişken.

GBD : Gelirlerin bugünkü değeri.

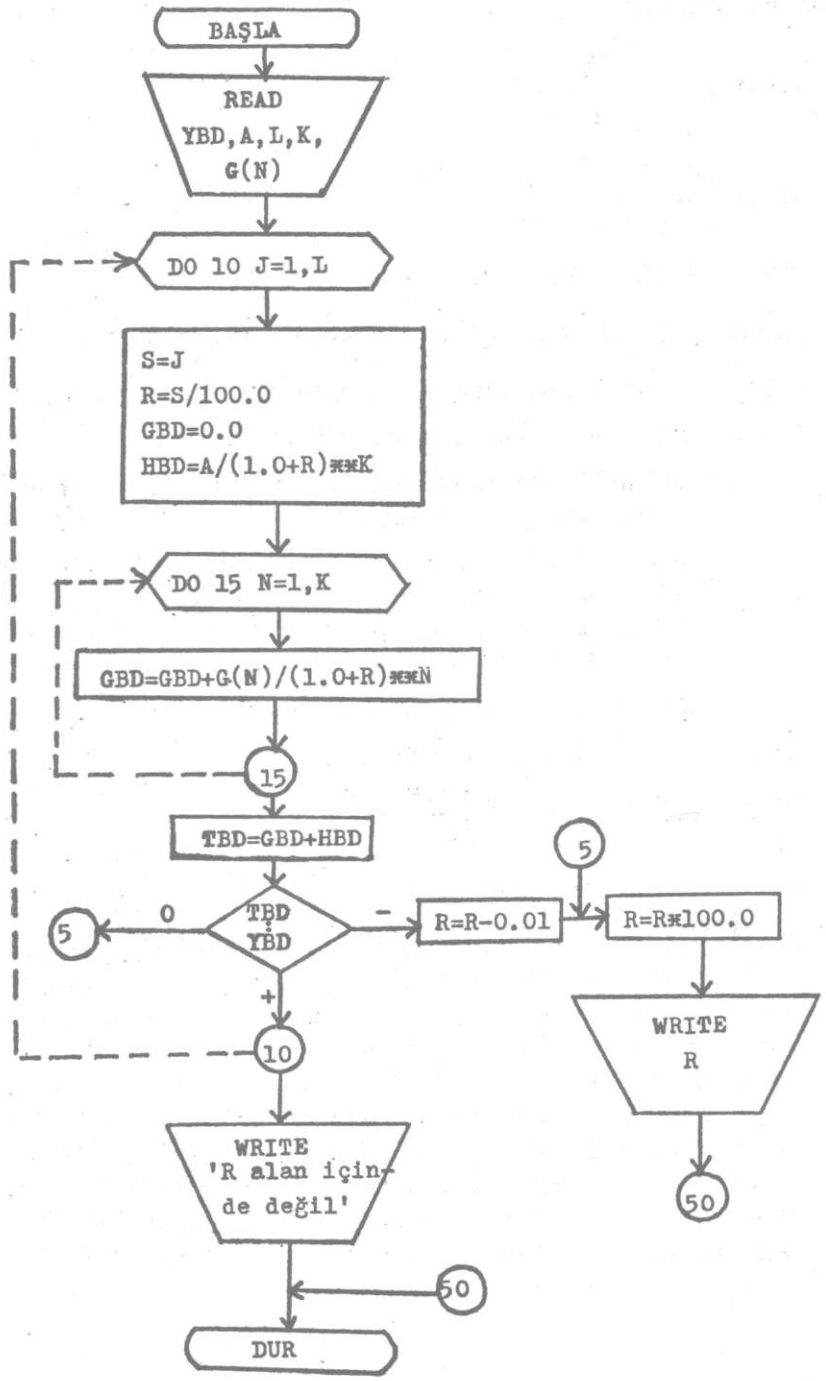
HBD : Hurdanın bugünkü değeri.

TBD : Toplam bugünkü değer.

K : Yatırımın yararlı ömrü.

Bitişik sayfada verilen akış-şeması ve programa göre programın girdileri arasında belirtilen L değişkeninin başlangıç değerini, 1 yerine tahmin edilen R oranına yakın olarak belirtmekle programda tekrar sayısı azaltılabilir. Örneğin R'nin % 25 olarak tahmin edildiği bir problemde L'nin başlangıç değeri 15 olarak alt bir sınır biçiminde belirtilebilir. Buna göre J=15, 16,...., 100 olarak sayılacaktır. J'nin 15'den başlatılması ile programın 15 kez tekrarıda önlenmiş oluyor. Kuşkusuz bu durumlar ancak r oranı hakkında bir kestirmede bulunabileceğimiz durumlar için söz konusudur. Eğer r oranına ilişkin bir kestirmede bulunamıyorsak bu durumda J başlangıç değeri 1 olarak alınacaktır.

Akış - Şeması


BAŞLA
READ
YBD,A,L,K,
G(N)
DO 10 J=1,L
S=J
R=S/100.0
GBD=0.0
HBD=A/(1.0+R)**K
DO 15 N=1,K
GBD=GBD+G(N)/(1.0+R)**N
15
TBD=GBD+HBD
TBD
YBD
0
5
-
+
R=R-0.01
5
R=R*100.0
WRITE
R
50
10
WRITE
'R alan için-
de değil'
50
DUR

Program : (Yararlı ömrü 40 yıl olan bir yatırım örnek olarak alınmıştır).

```
C  IC KARLILIK ORANININ BULUNMASI
      DIMENSION G(40)
      READ (5,7) YBD, A,L,K, (G(N), N=1,K)
    7 FORMAT (2F10.0, 2I4,/,(8F10.0) )
   25 FORMAT (5X, 'IC KARLILIK ORANI (YUZDE) =',F4.0)
   45 FORMAT (5X, 'R ALAN ICINDE DEGIL.L ALA
     1NINI DEGISTIR')
      DO 10 J=1,L
      S=J
      R=S/100.0
      GBD=0.0
      HBD=A/(1.0+R)**K
      DO 15 N=1,K
      GBD=GBD+G(N)/(1.0+R)**N
   15 CONTINUE
      TBD=GBD+HBD
      IF(TBD-YBD) 20,5,10
   20 R=R-0.01
    5 R=R*100.0
      WRITE (6,25) R
      GO TO 50
   10 CONTINUE
      WRITE (6,45)
   50 STOP
      END
```

Bu programda;

DO 10 J=1,L döngüsü L alanı içinde J'yi birer birer artırarak saymaktadır.

S=J deyimi ile J değeri gerçel sayıya dönüştürülmektedir.

R=S/100.0 deyimi ile S değeri yüzde oranına dönüştürülmektedir.

DO 15 N=1,K döngüsü yatırım ömrü kadar tekrar yaparak her R oranı için gelirlerin bugünkü değerini hesaplamaktadır.

R=R*100.0 deyimi ondalık sayı olarak bulunan R'yi ondalık biçiminden çıkarmaktadır. Örneğin, R=0.11 ise bu oran R=11 olacaktır. Yani ondalık R'yi tamsayıya dönüştüren bir düzeltme deyimidir.

III.

ALTPROGRAMLAR

Programlama çalışmalarında çeşitli problemlerin çözümü için algoritma (çözüm yöntemi) geliştirirken, belirli bazı işlemlerin bir kaç kez tekrarlanması gerekebilir. Bir program içinde belirli bazı işlemlerin tekrar tekrar yazılması programcı için arzu edilmeyen bir yüktür. Fakat tekrarlanan işlemler ana program içerisinde bir altprogram olarak düşünülür ve gerekli görülen yerde yapılacak işlemlerin altprogramın belirttiği işlemler olduğu bilgisayara iletilirse (yani altprogram çağrılırsa), altprogram işlemlerinin ana programda tekrar tekrar yazılması önlenebilir.

Diğer yandan altprogram olarak hazırlanan belirli işlemlerin ana programın istenilen noktalarında çağrıldıkları zaman tekrarlanması olanağı, başkaları tarafından daha önce yazılmış ve denenmiş olan programların altprogram olarak farklı ana programlarda kullanılmasını olası kılar. Ayrıca sık sık kullanılan bazı işlemler (örneğin, kare kök, logaritma, sinüs, kosinüs alma gibi işlemler) için hazırlanan programlar bilgisayar sistemlerinde altprogram olarak hazırlanır ve her programcı kullanabilir. Böylece, altprogramlar programlama çalışmalarında zaman açısından büyük yararlar sağlarlar.

Altprogramlar ana programdan bağımsız oldukları için ayrı derlenir ve hatalardan arındırılırlar. Bu ise büyük programlarda oldukça önemli derecede derleme kolaylığı sağlar. Çok büyük programları derleyip hatalardan arındırmak oldukça güç olabilir. Böyle durumlarda büyük programı altprogramlar halinde parçalara ayırıp derlemek ve hataları gidermek büyük kolaylık sağlar. Altprogram olarak derlenen ana program parçaları tekrar derleme gerektirmediği için derleme zamanı kısa olacaktır.

FORTRAN dilindeki altprogramlar genel olarak dört temel kümeye ayrılırlar: (1) Kütüphane fonksiyonları, (2) FUNCTION altprogramları, (3) Deyim fonksiyonları ve (4) SUBROUTINE altprogramları.

Bu altprogramlar ayrı ayrı alt başlıklar altında bundan sonraki sayfalarda açıklanacaktır.

3.1 Kütüphane Fonksiyonları

Bilgisayar sistemlerinde hazır olarak makinanın içinde bulunan bu altprogramlar çoğu kullanıcıların sık sık gereksinim duyduğu kare kök, logaritma, sinüs, kosinüs alma v.b belirli işlemleri yapmak için kullanılırlar. Bilgisayar sistemi içinde muhafaza edildikleri için kütüphane ya da arşiv fonksiyonları adını alan bu altprogramlar uygun kurallar çerçevesinde kullanılabilirler.

Örneğin, 192.6 sayısının kare kökünü hesaplamak isteyelim ve bulunacak sonuca da A diyelim. Bu durumda, kare kök kütüphane fonksiyonu,

A=SQRT (192.6)

biçiminde yazılır. SQRT fonksiyon adı kare kök sözcüğünün ingilizce karşılığı olan SQuare RooT) biçiminde anımsanabilir. Böylece yukarıdaki kütüphane fonksiyonuna göre 192.6 sayısının kare kökü olan sayı A değişkeni adı altında bellekte muhafaza edilecektir.

Kütüphane fonksiyonlarının ayraçları içinde bulunan ve işlemi yapılacak değerlere (192.6 sayısı gibi) programlama dilinde altprogramın "argümanı" adı verilmektedir.

Bir başka örnek olarak X değişkeninin sinüsünü bulmak istediğimizi varsayalım. Sinüs kütüphane fonksiyonu SIN,

SIN (X)

biçimini alacaktır. Fonksiyon sonucunu B değişkeni ile gösterirsek kütüphane fonksiyonunu kullanmak için

B=SIN (X)

yazılmalıdır. Böylece X'in sinüsü B değişkeni adı altında muhafaza edilecektir.

Genel kural olarak, kütüphane fonksiyonları altprogramlarının yazılış biçimi : eşitliğin sağ tarafında fonksiyonun adı ve ayraç içinde argüman ya da argümanlardan oluşmaktadır. Ancak bu biçim bazı kurallara uymak zorundadır. Belli başlı kütüphane fonksiyonlarını belirtmeden önce bu fonksiyonların yazılmasında uyulması gereken kuralları özetlemek uygun olacaktır.

1. Fonksiyonun argümanı bir sabit ya da bir değişken olabileceği gibi bir aritmetik ifade de olabilir. Örneğin;

```
A=SQRT (3.0★AC/245.0)
B=SQRT (VARYNS)
```

ifadeleri gibi.

2. Argüman sayısı ve türü fonksiyonun belirlenmesindeki argümanların sayısı ve türü ile uyuşum içinde olmalıdır. Örneğin, C=SQRT(A,B) gibi bir ifade yazılamaz. Çünkü SQRT fonksiyonu bir tek argümanın kare kökünü hesaplar. Diğer yandan C=SQRT (N) gibi bir ifadede yazılamaz. Çünkü fonksiyon adı gerçel bir değişken adıdır, çünkü S harfi ile başlamaktadır. Halbuki N argümanı bir tamsayı değişkenidir. Her fonksiyonun argüman sayısı ve türü bitişik sayfadaki fonksiyon tablosunda verilmiştir.

3. Kütüphane fonksiyonunun bulacağı değer fonksiyon adının ilk harfine bağlı olarak gerçel ya da tamsayılı bir sabit olacaktır. Dolayısiyle, K=SQRT (A) gibi bir ifade K bir tamsayı değişkeni olduğu için yazılmamalıdır.

4. Bir fonksiyon kendisinin bir argümanı olamaz. Bu kurala göre B=SQRT (3.0★SQRT (C)) biçiminde bir ifade geçerli değildir.

İşletmecilik alanında sık sık kullanılan bazı kütüphane fonksiyonları bitişik sayfadaki tabloda verilmiştir. Belirtilen kurallar ve tabloda verilen bilgiler çerçevesinde doğru olarak yazılmış bazı kütüphane fonksiyonları aşağıda verilmiştir.

Matematiksel İfade :	**FORTRAN İfadesi :**
$S=-\frac{1}{A}\ln t$	S= −1.0/A★ALOG (T)
$f=\lvert -3.0 \rvert \frac{\sin g}{\cos (a.b)}$	F=ABS (−3.0)★SIN (G)/COS(A★B)
$q=\sqrt{2a\frac{s}{t}}$	Q=SQRT (2.0★A★S/T)
$y=a^2 . e^a$	Y=A★★2★EXP (A)
$r=\log y + \log z$	R=ALOG10 (Y) +ALOG10 (Z)

BAZI KÜTÜPHANE FONKSİYONLARI

Fonksiyon Adı	FORTRAN Adı	Matematiksel İfade	FORTRAN İfadesi	Argüman Sayısı	Argüman Türü	Fonksiyon Türü
Kare Kök	SQRT	$y=\sqrt{x}$	Y=SQRT (X)	1	Gerçel	Gerçel
	DSQRT		Y=DSQRT (X)	1	Çift ha.	Çift ha.
Doğal Logaritma	ALOG	$y=\ln x$ $(\log_e x)$	Y=ALOG (X)	1	Gerçel	Gerçel
Adi Logaritma	ALOG10	$y=\log x$	Y=ALOG10 (X)	1	Gerçel	Gerçel
Üssel	EXP	$y=e^x$	Y=EXP (X)	1	Gerçel	Gerçel
Trigon. Sinüs	SIN	$y=\sin x$	Y=SIN (X)	1	Gerçel	Gerçel
Trigon. Kosinüs	COS	$y=\cos x$	Y=COS (X)	1	Gerçel	Gerçel
Trigon. Tanjant	TAN	$y=\tan x$	Y=TAN (X)	1	Gerçel	Gerçel
Mutlak Değer	ABS	$y=\lvert x \rvert$	Y=ABS (X)	1	Gerçel	Gerçel
	IABS	$k=\lvert n \rvert$	K=IABS (N)	1	Tamsayı	Tamsayı
Modülo[1]	MOD	$n=K \pmod m$	N=MOD (K,M)	2	Tamsayı	Tamsayı
	AMOD	$a=Y \pmod Z$	A=AMOD (Y,Z)	2	Gerçel	Gerçel

(a) Tabloyu karmaşıklaştırmamak için fonksiyonların çift hassaslıklı biçimleri verilmemiştir. Ancak fonksiyonların gerçel biçimlerinin önüne çift hassaslık simgesi D'yi eklemek yeterli olacaktır.

(1) n=K (mod m) ifadesinde K değeri m'ye bölünür kalan sayı n olarak kaydedilir. Örneğin, K=13 ve m=5 ise n= 3 olacaktır. Modülo fonksiyonu kendi kendini yineleyen bir aritmetik ifade aracılığı ile benzetim (simulasyon) modellerinde rassal (tesadüfi) sayı üretiminde sık sık kullanılır.

$t=\ln a+\sqrt{45}$	T=ALOG (A)+SQRT (45.0)
$X=\log (a+b/c)$	X=ALOG10 (A+B/C)

Aşağıdaki kütüphane fonksiyonları ise hatalı yazılmışlardır.

a) Y=ALOG10 (Y,Z)+SQRT (Y)

b) B=COS (X)+SUN (Y)

c) F=SQRT (45)★42.3

d) N=IABS (−27.40)

e) S=EXP (B/C)+SQRT (3.4★SQRT (9.0))

Hatalar :

a) ALOG10 fonksiyonu yalnızca bir tek argüman için belirlenmiştir. İki argüman Y ve Z birlikte olamaz.

b) Sinüs fonksiyonu SUN değil SIN biçimindedir.

c) SQRT fonksiyonu gerçel değerli halbuki argümanı 45 bir tamsayı sabitidir.

d) IABS fonksiyonunun argümanı tamsayılı olmalıdır. −27.40 argümanı gerçeldir. Bu sayının mutlak değerini bulmak için ABS fonksiyonu kullanılmalıdır.

e) Bir fonksiyon kendisinin argümanı olamaz. Ancak verilen ifadede A=SQRT (9.0) olarak belirlenirse ve S=EXP (B/C)+SQRT (3.4★A) olarak yazılabilir.

Buraya kadar açıklanmaya çalışılan kütüphane fonksiyonları hemen hemen tüm bilgisayar sistemlerinde mevcuttur. Ancak bazı sistemlerde iki ya da daha fazla sayıda argümanın en küçük ve en büyük değerlerini bulmak için bir grup kütüphane fonksiyonu bulunmaktadır. İşletmecilik alanında sık sık başvurulan bu fonksiyonlar şunlardır :

Fonksiyon Adı :	FORTRAN Adı :	Argüman Türü	Fonksiyon Türü	Argüman Sayısı
En Büyük Değer (Maksimum)	AMAXO	Tamsayı	Gerçel	⩾ 2
	AMAX1	Gerçel	Gerçel	
	DMAX1	Çift has.	Çift has.	
	MAXO	Tamsayı	Tamsayı	
	MAX1	Gerçel	Tamsayı	
En Küçük Değer (Minumum)	AMINO	Tamsayı	Gerçel	⩾ 2
	AMİN1	Gerçel	Gerçel	
	DMIN1	Çift has.	Çift has.	
	MINO	Tamsayı	Tamsayı	
	MIN1	Gerçel	Tamsayı	

Görüldüğü gibi en büyük ve en küçük değerleri bulmak için kullanılan fonksiyonların en az iki argümanı bulunmalıdır. Örneğin,

A,B,C,D gerçel değişkenlerinin en büyüğünü bulmak için AMAX1 kütüphane fonksiyonu AMAX1 (A,B,C,D) olarak yazılır. AMİN1 fonksiyonuda AMİN1 (A,B,C,D) biçimi ile söz konusu değişkenlerin en küçüğünü bulacaktır. MAX1 ve MIN1 fonksiyonları da aynı işlemi yapacaklardır. Ancak bu kez sonuçlar (en büyük ve en küçük değerler tamsayılı sayılar olarak muhafaza edilecektir.

3.2 Aritmetik Deyim Fonksiyonları

Bilgisayarda hazır olarak bulunan kütüphane fonksiyonlarından farklı olarak programcı kendi programında sık sık kullandığı matematiksel bir ifadeyi bir fonksiyon alt programı olarak tanımlayabilir. ve ana programın değişik yerlerinde çağırıp kullanabilir. Tek bir aritmetik ifadeden oluşan bu altprogram türü ana program ile birlikte derlenir ve bu nedenle de başka ana programlarda çağrılamaz.

Deyim fonksiyonlarının genel kalıbı,

$$f\ (d_1,d_2,d_3,....,d_n) = e$$

biçimindedir. Burada;

f : Fonksiyonun adını belirtecek olan bir FORTRAN değişkeni,

d_n : Fonksiyonun temsili argümanı ya da fonksiyonun aralarındaki ilişkiyi belirlediği değişkenler,

e : Dizinli değişken ihtiva etmeyen bir aritmetik ifadeyi temsil etmektedir.

Örneğin, bir programda bir çok yerlerde $A=S(1+r)^n$ biçimindeki bileşik faiz formülünün kullanılması gerektiğini varsayalım ve bunu bir deyim fonksiyonu olarak tanımlayalım. Fonksiyona BILFAZ adını verelim. Yukarıda verilen kalıba göre fonksiyon,

$$BILFAZ(S,R,N) = S*(1.0+R)**N$$

olarak belirlenebilir. Bu ifade bir aritmetik atama deyimi değildir. Başka bir anlatımla eşitliğin sağ tarafındaki ifade değerlendirilip sol taraftaki simgeler için muhafaza edilmeyecektir. Bu ifadenin belirtmek istediği : BILFAZ değişkeninin, üç (S,R,N) değişken (ya da temsili argüman) içeren ve bu değişkenler arasındaki ilişkiyi değerlendiren bir fonksiyon olduğudur. Bu şekilde tanımlanmış bir fonksiyondan sonra programda BILFAZ fonksiyonu görüldüğünde bilgisayar S*(1.0+R)**N ifadesinin belirttiği işlemi yapacaktır.

Örneğin programda,

T=BILFAZ(PO,RO,I1)

ifadesi ile karşılaşılırsa BILFAZ(S,R,N) fonksiyonunda S değişkeni yerine PO,R yerine RO ve N yerine I1 değişkenlerinin sayısal değerleri konularak S✱(1.0+R)★★N ifadesinin belirttiği işlem yapılır ve sonuç T değişkeni adı altında depolanır. Yani yukarıdaki ifade,

T=PO✱(1.0+RO)★★I1

biçiminde yazılmış gibi değerlendirilir. Fonksiyondaki S,R,N argümanları temsili argümanlar, daha sonra fonksiyonu çağırma ifadesinde kullandığımız PO,RO,I1 değişkenleri ise gerçek argümanlar olarak adlandırılırlar. Görüldüğü gibi bir aritmetik fonksiyonu programın herhangi bir noktasında çağırmak için fonksiyonun gerçek argümanlar ile birlikte eşitliğin sağ tarafında belirtilmesi yeterli olacaktır. Böylece,

BILFAZ(S,R,N) =S✱(1.0+R)★★N

biçiminde tanımlanmış bir deyim fonksiyonu

T=BILFAZ(5000.0,0.05,10)

T=BILFAZ(A,F,L)

T=BILFAZ(A+C, G—0.05, K)

Y=BILFAZ(T,R,M)★Z/A

biçiminde yazılmış ifadeler ile çağrılabilir. Her çağıran ifadesinin gerçek argümanları fonksiyonun temsili argümanlarının yerini alır ve belirtilen işlem yapılır. Örneğin üçüncü çağıran ifadesinde : (A+C) S'nin, (G—0.05) R'nin ve K ise N temsili argümanlarının yerlerini alacaklardır. Yine görüldüğü gibi son ifadede T,R,M gerçek argümanları temsili argümanların yerini alıp S✱ (1.0+R)★★N yani T★ (1.0+R)*★M işlemi yapıldıktan sonra bulunan değer Z ile çarpılacak ve A'ya bölündükten sonra Y değişkeni adı altında depolanacaktır.

Bundan önceki açıklamalarda örnek olarak alınan bileşik faiz formülü;

BILFAZ(S,N) =S✱(1.0+R)★★N

olarak da fonksiyon biçiminde tanımlanabilir. Burada farklılık aritmetik işlemdeki R değişkeni eşitliğin sol tarafında fonksiyonun temsili argümanı olarak gösterilmemiştir. R gibi aritmetik ifadede yer alan ancak fonksiyonun temsili argümanı olarak belirtilmeyen değişkenlere

fonksiyonun parametreleri denir. Yukarıdaki gibi tanımlanan bir fonksiyon;

```
Y=BILFAZ(T,M)*Z/A
```

biçimindeki bir ifade ile çağrılabilir. Böylece gerçek argümanlar (T,M) temsili argümanlar ise (S,N)'dir.

Bir başka örnek olarak,

$$x^3+2x^2+3x+1$$

gibi bir işlemi deyim fonksiyonu olarak tanımlayalım. Mevcut tek değişkeni temsili argüman olarak alır ve fonksiyon adıda Y olarak seçilirse,

```
Y(X)=X**3+2.0*X**2+3.0*X+1.0
```

biçiminde bir Y(X) deyim fonksiyonu yazılabilir. Programın daha sonraki bölümlerinde bu fonksiyon istenilen yerde çağrılabilir. Örneğin;

```
W=2.0
Z=Y(W)
```

olarak çağrılırsa Y(X) fonksiyonunda temsili argüman X yerine W gerçek argümanının değeri olan 2.0 konulacak ve Z değişkeninin değeri,

```
Z=2.0**3+2.0*2.0**2+3.0*2.0+1.0
Z=8.0+2.0*4.0+3.0*2.0+1.0
Z=8.0+8.0+6.0+1.0
Z=23.0
```

olarak bulunur.

Deyim fonksiyonları dizinli değişkenlerle karıştırılmamalıdır. Dizinli değişkenler devamlı olarak DIMENSION deyimi ile program başında belirtilirler. Deyim fonksiyonu programda işlenebilir ilk deyimden önce yer almalıdır.

Şimdiye kadar verilen açıklamaların da belirtebileceği gibi deyim fonksiyonlarının tanımlanmasında aşağıdaki kurallara uyulmalıdır.

1. Temsili argümanlar ile gerçek argümanlar sayı ve değişken türü bakımından aynı olmalıdır. Örneğin;

```
GEL(I,S)=P**I+3.0*S
```

olarak tanımlanan bir fonksiyon

```
Y=GEL(D,N)
```

biçiminde çağrılamaz. Çünkü I ile D ve S ile N farklı türden argümanlardır. Eğer deyim fonksiyonu

```
Y=GEL(N)
```

şeklinde çağrılırsa yine yanlış olacaktır. Çünkü fonksiyondaki temsili argüman sayısı 2 olmasına karşın fonksiyon çağrıldığında belirtilen gerçek argüman sayısı 1'dir. Bu nedenlerle doğru bir çağırma ifadesi,

```
Y=GEL(N,D)
```

biçiminde olacaktır. Çünkü temsili ve gerçek argümanlar sayı ve tür bakımından aynıdır.

2. Deyim fonksiyonunun türü fonksiyon adının ilk harfine göre belirlenir. Örneğin, GEL(I,S) fonksiyonu gerçel olduğundan fonksiyonun değeri gerçel bir sayı olacaktır.

3. Gerçek argümanlar bir aritmetik ifade de olabilir.

Örneğin;

```
Y(X)=X**2+X/4.0
```

fonksiyonu,

```
T=Y(D*2.0)
```

biçiminde çağrılabilir. Burada T değişkeninin değeri,

```
T=(D*2.0)**2+(D*2.0)/4.0
```

ifadesi yazılmış gibi hesaplanacaktır.

4. Deyim fonksiyonundaki aritmetik ifadede dizinli değişken bulunmamalıdır. Örneğin;

```
ADA(X)=X(I)**3
```

gibi tanımlanan fonksiyonda X değişkeni dizinli olduğu için bu deyim fonksiyonu geçerli değildir.

3.3 FUNCTION Altprogramları

Bir programın yazılması sırasında fonksiyon olarak tanımlanmak istenen ifade çoğu zaman bir tek matematiksel ifade biçiminde deyim fonksiyonu olarak yazılmayacak kadar karmaşık olabilir. Arzu edilen altprogram bir tek matematiksel ifade ile belirlenemeyecek biçimde ise, deyim fonksiyonu yerine FUNCTION altprogramı olarak tanımlanır.

FUNCTION altprogramı da deyim fonksiyonu gibi ana programa çağrıldığı zaman bir tek değer getirir. FUNCTION altprogramı ana programdan bağımsız kendi başına bir program olup bir END deyimi de kapsar. Deyim fonksiyonu ana programın işlenebilir ilk deyiminden önce yer almasına karşın FUNCTION altprogramı ana programın en son deyiminden (yani END deyimi) sonra yer alır. Bir FUNCTION altprogramının belirlenmesinde aşağıdaki kurallar göz önüne alınmalıdır.

1. Fonksiyon altprogramının ilk ifadesi,

FUNCTION $f(d_1,d_2,.....,d_n)$

genel kalıbı biçiminde olmalıdır. Burada,

FUNCTION : Altprogramın bir fonksiyon olduğunu belirtir ve olduğu gibi yazılmalıdır.

f : Fonksiyonun bir FORTRAN değişkeni olan adını belirtir.

$d_1,...,d_n$: Fonksiyonun kapsadığı temsili argümanları göstermektedir. Temsili argümanlar dizinli olmayan değişken, dizi ya da bir başka altprogram adı olabilirler.

2. Fonksiyonun adı en az bir kez bir aritmetik atama deyiminin solunda değişken adı olarak yer almalıdır. Başka bir anlatımla, f=e biçiminde bir atama deyimi FUNCTION altprogramında yer almalıdır. Burada f fonksiyonun adını e ise bir aritmetik ifadeyi temsil etmektedir. Fonksiyon adı ana programa çağrılmak istenen değere göre belirlenir. Eğer fonksiyondan ana programa çağrılacak değer gerçel bir sabit ise fonksiyon adı gerçel, bir tamsayı sabiti ise fonksiyonun adı bir tamsayı değişkeni olmalıdır.

3. FUNCTION altprogramı ana programa deyim fonksiyonu gibi çağrılır. Yani, $f(d_1,d_2,d_3,.,..,d_n)$ ifadesi ana programda eşitliğin sağ tarafında yer alır. Ancak bu kez $d_1,d_2,d_3,...,d_n$ argümanları gerçek argümanlardır ve çağırma sırasında temsili argümanların yerini alırlar. Deyim fonksiyonunda olduğu gibi FUNCTION altprogramında da temsili argümanlar ile gerçek argümanlar sayı ve tür bakımından aynı olmalıdır. Çağıran ifadesi ile kontrol FUNCTION altprogramına aktarıldığında çağıran ifadesindeki gerçek argümanlar fonksiyon altprogramındaki temsili argümanların yerini alır ve altprogramın belirttiği işlemler yapılır. Fonksiyon adının bir değişken olarak eşitliğin sol tarafında yer aldığı en son f=e biçimindeki ifade de f değeri fonksiyon altprogramının bulduğu değer olacaktır.

4. FUNCTION altprogramından kontrolu ana programa aktarmak için fonksiyon altprogramı bir RETURN deyimi kapsamalıdır. Altprogramda birden fazla RETURN deyimi bulunabilir. Bilgisayar FUNCTION altprogramında RETURN deyimi ile karşılaştığı an kontrolu ana programa aktarır ve ana programda kaldığı noktadan işleme devam eder.

Ana programdan bağımsız olarak derlenen FUNCTION altprogramında derleme sonunu belirtmesi için altprogram bir END deyimini en son deyim olarak ihtiva etmelidir. Böylece altprogramda derleme sonunu END ve işlem sonunu RETURN deyimi belirtmektedir. Yani altprogramda STOP deyimi bulunmamasına karşın işlem RETURN deyimi ile bitirilmektedir.

Bu açıklama ve kurallara göre bir FUNCTION altprogramının genel kalıbı aşağıdaki biçimi almaktadır :

```
FUNCTION f(d1,d2,d3',....,dn)
:
:
RETURN
:
END
```

Örnek 1 : Kalite Kontrol Problemi

Bir makinanın üretmiş olduğu mamüllerin ortalama % p kadarı kusurlu çıkmaktadır. Bu ürünlerden (n) kadarı kontrol edilirse (k) ya da daha az sayıda kusurlu mamül bulunmanın olasılığı p(k) nedir? Kalite kontrol problemlerinde sık sık sorulan bu soruyu yanıtlamak için baş vurulan formül,

$$p(k) = p^k (1-p)^{n-k} \frac{n!}{k!\,(n-k)!}$$

biçimindeki binomial olasılık dağılım formülüdür. Çünkü kontrol olayı bir Bernoulli olayıdır. Yani kontrol edilen mamül ya kusurlu ya da sağlamdır.

Yukarıdaki formülde p(k), k sayıda kusurlu mamül bulunma olasılığıdır. k'dan daha az olan sayıların olasılıklarınıda göz önüne almak için P(0,1,2,3,...,k) biçiminde k ve k'dan küçük olan değerlerin olasılıklarının toplamı olarak göz önüne alınmalıdır, yani;

$$P(K) = \sum_{i=0}^{k} p(i) \qquad \text{gibi.}$$

Böylece hiç kusurlu mamül bulmamadan (sıfırdan) başlayıp k'ya kadar olan olasılıkları yukarıda verilen formüle göre tek tek hesaplar ve bulunan değerleri toplarsak yapacağımız hesaplama bitmiş olacaktır. Görüldüğü gibi her hesaplamada biri n! biri k! ve biri de (n—k)! için üç kez faktöryel hesaplamak gerekmektedir. O halde n gibi bir sayı için n!=1x2x3x.....xn biçiminde faktöryeli hesaplayacak bir FUNCTION altprogramı yazar ve faktöryel hesaplamak gereken her noktada bu altprogramı çağırırsak aradığımız olasılığı hesaplayacak program hazırlanmış olacaktır.

Programda kullanılacak değişkenleri aşağıdaki gibi tanımlayalım:

NFAKT : Faktöryeli hesaplayacak olan FUNCTION altprogramının adını temsil eden bir tamsayı değirkeni.

NF : Hesaplanacak olan n! değeri.

KF : Hesaplanacak olan k! değeri.

NKF : (n—k)! sonucunu temsil eden değişken.

P : Kusurlu mamül bulunma olasılığı.

PK : k ya da daha az sayıda kusurlu mamül bulma olasılığı.

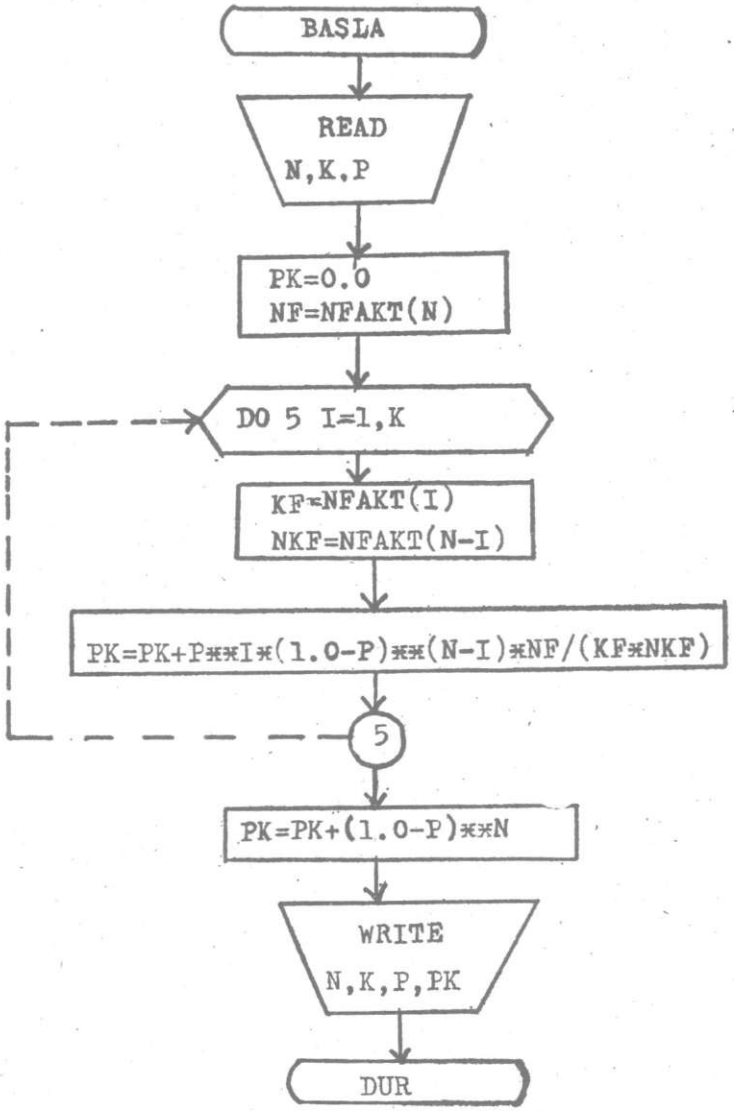
Bu değişkenleri kullanarak ana programın akış şemasını bitişik sayfadaki gibi çizebiliriz.

Ana programın akış şemasında NF=NFAKT(N) ifadesi NFAKT adını taşıyan bir fonksiyon altprogramının çağrıldığını ve bu altprograma N gerçek argümanının aktarıldığını belirtmektedir. Ayrıca bu ifade altprogramdan N gerçek argümanı için getirilen değerin NF değişkeni adı altında muhafaza edileceğini belirtmektedir. Başka bir deyişle bu bir fonksiyon çağırma ifadesidir.

DO döngüsü içinde yer alan KF=NFAKT(I) ve NKF=NFAKT (N—I) deyimleri de fonksiyon çağırma ifadeleri olup K'ya kadar I'nin her artışında I! ve (N—I)! değerlerini FUNCTION altprogramından getirmektedir. Döngü tamamlandığı zaman bulunan PK değeri 1'den K'ya kadar olan kusurlu mamül bulunma olasılıklarının toplamını belirtmektedir. Ancak kusurlu mamül bulunmama (sıfır kusurlu) olasılığını kapsamamaktadır. Çünkü I=1,2,...,K ya kadar giderken 1'den başlamaktadır. Böyle olunca PK değerine bu olasılık miktarıda eklenmelidir.

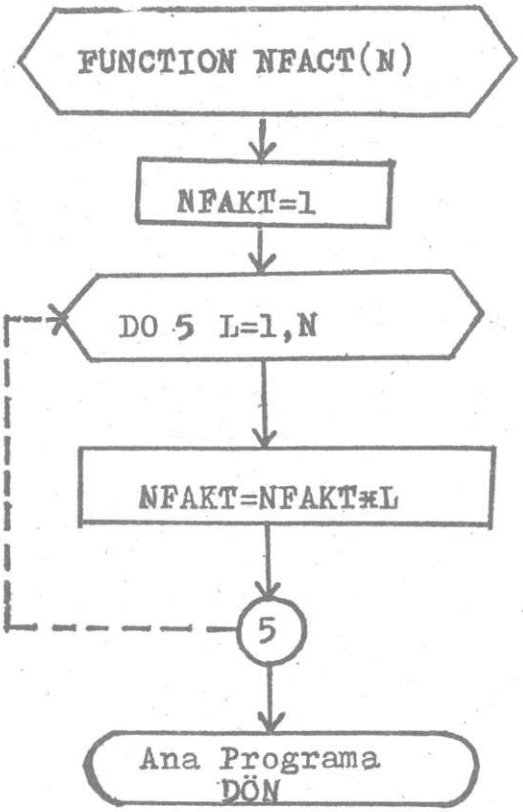
Kusurlu mamül bulunmama olasılığı ise başlangıçta verilen formüle göre ($p^0=1$ ve $o!=1$ olduğu için),

$$p(0) = (1-p)^n \qquad \text{dir.}$$



Bu değeride (1.0—P)★★N olarak PK'nın DO döngüsü sonunda elde edilen değerine eklersek en son PK değeri bulunur. Bu ise aranılan yanıttır. Yani K ya da daha az kusurlu mamül bulunma olasılığıdır.

Faktöryeli hesaplayacak altprogramının akış şeması da aşağıdaki gibi çizilebilir.



Akış şemasındaki ilk şekil FUNCTION NFAKT altprogramını N temsili argümanı ile belirtmektedir. Fonksiyon adı NFAKT bir değişken olarak eşitliğin sol tarafında bir atama deyiminde bulunmaktadır, yani NFAKT=NFAKT★L. Bu ifade aynı zamanda faktöryel değerlerini hesaplamaktadır. Kontrol altprogramdan ana programa dönerken NFAKT'ın en son değeri altprogram değeri olarak anaprograma gönderilmekte ve gerekli değişkenlere (NF,KF ve NKF) verilmektedir. Ana ve altprogram akış şemalarına dayalı olarak tüm program aşağıdaki gibi yazılabilir.

Ana Program :

```
C  KALITE KONTROL VE BINOMIAL DAGILIM
C
       READ (5,10) N,K,P
   10  FORMAT (2I2, F4.2)
       PK=0.0
```

```
      NF=NFAKT(N)
      DO 5 I=1,K
      KF=NFAKT(I)
      NKF=NFAKT(N-I)
      PK=PK+P**I*(1.0-P)**(N-I)*NF/(KF*NKF)
    5 CONTINUE
      PK=PK+(1.0-P)**N
      WRITE (6,15)N,K,PK
   15 FORMAT (5X,/ /,I2, 3X, 'MAMULDE', I2, 2X, 'YA DA
     1DAHA AZ KUSURLULUK OLASILIGI=', F4.2)
      STOP
      END
C
C  ALTPROGRAM
C
      FUNCTION NFAKT(N)
      NFAKT=1
      DO 5 L=1,N
      NFAKT=NFAKT*L
    5 CONTINUE
      RETURN
      END
```

Verilen programda görüldüğü gibi FUNCTION NFAKT altprogramının temsili argümanı N'dir. Altprogram çağrıldığı zaman ana programdaki çağıran ifadelerinin gerçek argümanları bu temsili argümanın yerini alacaklardır. Örneğin, ilk çağıran NF=NFAKT (N) ifadesinde gerçek argüman da yine N'dir. Altprogram bağımsız olduğu için bu bir karışıklığa yol açmayacaktır. Yani bilgisayar bu iki argümanı ayrı ayrı görecektir. Böylece N'nin gerçek (okutulan) değeri altprogramda temsili argüman N'nin yerini alacak ve hesaplanan NFAKT değeri ana programa RETURN deyimi aracılığı ile gönderilecek ve NF değişkeninin değeri olarak depolanacaktır.

Daha sonra kontrol ana programın DO döngüsüne geçmektedir. Bu döngüde ilk iki deyim yine altprogramı çağıran ifadelerdir. Yapılacak işlemler aynıdır. Ancak ilk ifadede gerçek argüman I ikinci ifadede N—I dır. Dolayısiyle FUNCTION altprogramına bu değerler aktarılacak ve işlemler yapılacaktır. Örneğin, K=4 ve N=8 olarak veriliyorsa ve döngünün üçüncü tekrarında yani I=3 iken (N—I) gerçek argümanının değeri 8—3=5 olacaktır. Altprograma I=3 değeri aktarıldığı zaman KF ve N—I=8—3=5 aktarıldığı zaman da NKF değer-

leri hesaplanacaktır ve bu değerler izleyen ifadede yerlerine konulup PK değeri hesaplanmaktadır.

FUNCTION altprogramında görüldüğü gibi fonksiyon adı değişken olarak iki kez eşitliğin sol tarafında yer almaktadır ve aldığı en son değer ana programa gönderilmektedir. NFAKT=1 ifadesi faktöryel hesaplamada başlangıç değerini oluşturmak için kullanılmıştır. Daha önceki örneklerde olduğu gibi eğer NFAKT=0 olarak belirlenirse tüm işlem sonucu sıfır olacaktır. Bu nedenle 1 olarak belirlenmiştir. Ana programlara dönüşü daha öncede belirtildiği gibi RETURN deyimi sağlamaktadır. Ayrıca programda fark edilebileceği gibi CONTINUE deyimleri hem ana programda hem de altprogramda aynı numarayı (5) taşımaktadır. Ancak daha öncede açıklandığı gibi bu karışıklık yaratmaz. Çünkü altprogram ana programdan bağımsız olarak işlenmektedir.

Örnek 2 : Bir Dizinin En Küçüğünü Bulma

Bir S(N) dizisinde en küçük elemanın değerini belirleyecek bir program, FUNCTION altprogramı kullanılarak hazırlanmak isteniliyor. Bu işlevi görecek altprogram aşağıdaki gibi yazılabilir.

```
      FUNCTION ENKUC (S,N)
      DIMENSION S (N)
      ENKUC=S (1)
      DO 10 I=2,N
      IF (S (I).GE.ENKUC) GO TO 10
      ENKUC=S (I)
10    CONTINUE
      RETURN
      END
```

Yukarıdaki altprogramda dizinin elemanları başlangıçta ENKUC değişkenine atanan S(1) elemanı değeri ile karşılaştırılmaktadır. Her aşamada dizinin bundan küçük elemanları ENKUC değişkenine tekrar atanmaktadır. Son aşamada ENKUC değişkeninin değeri dizinin en küçük elemanı olmaktadır.

Şimdi bu altprogramı bir işletmenin bir aylık dönemdeki günlük satış hasılatları (GUNHAS) arasında en küçük satış hasılatını (EKHAS) belirlemek için yazacağımız bir ana programda kullanalım.

Ana Program :

```
      DIMENSION GUNHAS (30)
      READ (5,10) (GUHAS (I), I=1,30)
10    FORMAT (10F8.2)
      EKHAS=ENKUC (GUNHAS, 30)
      WRITE (6,15) EKHAS
15    FORMAT (5X, 'EN KUCUK HASILAT=', F10.2)
      STOP
      END
```

Böylece yazılmış olan ana programa önceden belirtilen ENKUC fonksiyon altprogramı eklenirse, yani altprogram ana programın END deyiminden sonra ana programa eklenirse istenilen en küçük günlük satış hasılatı (EKHAS) belirlenip çıktıya yazılacaktır.

Görüldüğü gibi ana programda ENKUC fonksiyon altprogramı çağrıldığı zaman çağıran EKHAS=ENKUC (GUNHAS, 30) ifadesindeki gerçek argümanlar fonksiyon altprogramındaki temsili argümanların yerini alacaktır. Yani GUNHAS S 'nin ve 30'da N'nin yerini almaktadır. GUNHAS ve S argümanlarının her ikiside dizinli oldukları için her iki programda boyutları DIMENSION deyimi ile belirtilmektedir. Hemen belirtilmelidir ki her ikisinin boyutları bir birine eşit olmalıdır. Örneğimizde boyutlar değişken olan argümanlarla programlar arasında ilişkilendirildiği için bu eşitlik kuralı kolaylıkla sağlanmaktadır. Altprogramda en küçük değer eşitliğin sol tarafında yer alan fonksiyon adı ENKUC ile ana programa iletilmektedir. Böylece RETURN deyimi gereği ana programa dönüldüğünde EKHAS değişkeni, fonksiyon adı ENKUC değişkeninin aldığı değeri almaktadır.

3.4 SUBROUTINE Altprogramları

Bu alt programlar ile FUNCTION altprogramları arasında büyük benzerlikler vardır. Şöyle ki :

1. Her iki altprogram türü de ana programdan bağımsız olarak derlenen ve ana programdan ayrı olan altprogramlardır.

2. Her ikiside FORTRAN kurallarına göre oluşturulan bir ad taşırlar.

3. Bu her iki altprogramda da temsili argümanlar ile en az bir tane RETURN deyimi ve en son deyim olarak END deyimi bulunur.

Bu benzerliklere karşın her iki program arasında önemli farklılıklar vardır. SUBROUTINE altprogramlarını açıklama amacı ile bu farklılıklar aşağıdaki gibi sıralanabilir.

1. Fonksiyon altprogramlarının ilk ifadesinde FUNCTION deyimi bulunmasına karşılık ikincisinde SUBROUTINE deyimi kullanılır.

2. SUBROUTINE altprogramının adı ile altprogramın türü arasında bir ilişki yoktur. Çünkü bu altprogram adı altprogram içinde her hangi bir değer almaz. Halbuki FUNCTION altprogramının adı altprogram içinde en az bir defa eşitliğin sol tarafında bulunur, yani bir değer alır. Böyle olunca SUBROUTINE altprogramı içinde altprogramın adı bir argüman, değişken ya da bir dizi adı olarak kullanılmamalıdır. Aynı nedenlerle SUBROUTINE altprogramının hesaplayacağı değer ile bu altprogramın adı arasında da bir ilişki yoktur ve aynı türden olma zorunluluğu bulunmamaktadır.

3. Bir FUNCTION altprogramı ana programa genellikle bir tek değer götürdüğü halde SUBROUTINE altprogramı bir ya da birden fazla değeri ana programa götürür. Bu götürme işlemi argümanlar aracılığı ile olur.

4. Hiç argümanı olmayan bir SUBROUTINE altprogramı olabilir. Halbuki bir FUNCTION altprogramının en az bir argümanı bulunmalıdır.

5. Bir SUBROUTINE altprogramı ana programda bu altprogramın adını ve argümanları belirten bir CALL deyimi ile çağrılır. Halbuki FUNCTION altprogramı bu deyimi gerektirmez ve ana programda adı geçtiği zaman çağrılır.

Fonksiyon altprogramları ile olan benzerlikleri ve farklılıkları açısından temel özellikleri belirtilen SUBROUTINE altprogramlarının genel biçimi aşağıdaki gibidir.

```
SUBROUTINE f (d1,d2,d3,.....,dn)
:
:
RETURN
:
END
```

Ana programdaki çağıran ifadesi ise,

```
CALL f (a1,a2,a3,.....,an)
```

biçimindedir. Burada yine f harfi SUBROUTINE altprogramının adını, $d_1,\ldots,d_n$ altprogramdaki temsili argümanları ve $a_1,\ldots,a_n$ ise ana prog-

ramdaki gerçek argümanları belirtmektedir. CALL deyimi ise altprogramın çağrılacağını ifade etmektedir.

SUBROUTINE altprogramlarındaki temsili argümanlar girdi, çıktı ve karışık argümanlar olarak üç kümeye ayrılabilirler. Girdi argümanları ana programdan altprograma veri sağlarlar ve değerleri altprogram içindeki ifadeler tarafından değiştirilmez. Çıktı argümanları ise değerleri altprogram içinde hesaplanır ve ana programa götürülürler. Karışık argümanlar da altprograma veri sağlar, değerleri altprogram içindeki ifadeler tarafından değiştirilebilir ve ana programa götürülebilirler.

Örnek 1 :

Konuya daha çok açıklık getirmek için fonksiyon altprogramı ile ilgili olarak verilen ikinci örneği, yani bir aylık dönemdeki günlük satış hasılatları (GUNHAS) arasında en küçük değerli satış hasılatını (EKHAS) belirleyecek bir programı SUBROUTINE altprogramı kullanarak yazalım.

Daha önceki örnekte 30 günlük satış hasılatları arasında EKHAS bulunurken bunun hangi güne ait olduğu yani dizinin hangi elemanı olduğu bilinmiyordu. Bu kez bulunan EKHAS değerinin dizinin kaçıncı sırasındaki elemanı olduğunu da bilmek isteyelim ve bunu da NO değişkeni ile gösterelim. Eşit değerli eleman bulunduğu zaman sıra numarası en büyük olan elemanı alalım. SUBROUTINE adına da bu kez KUCUK diyelim.

Ana Program :

```
      DIMENSION GUNHAS (30)
      READ (5,10) (GUNHAS (I), I=1,30)
10    FORMAT (10F8.2)
      CALL KUCUK (EKHAS, NO, GUNHAS, 30)
      WRITE (6,15) EKHAS, NO
15    FORMAT (5X, 'EN KUCUK HASILAT=', F8.2,/, 5X,
     ★'SIRA NUMARASI=', I3)
      STOP
      END
```

Altprogram :

```
      SUBROUTINE KUCUK (ENKUC, L,S,N)
      DIMENSION S(N)
      ENKUC=S(1)
      DO 10 I=2,N
      IF (S(I). GT.ENKUC) GO TO 10
      ENKUC= S(I)
      L=I
   10 CONTINUE
      RETURN
      END
```

Yukarıda verilen ana programda altprogram,

```
CALL KUCUK (EKHAS, NO, GUNHAS, 30)
```

ifadesi ile çağrılmaktadır. Bu ifadede EKHAS, NO, GUNHAS ve 30 ana-programın gerçek argümanları olup kontrol altprograma geçtiği zaman altprogramın temsili argümanlarının yerini belirtilen sıraları ile alacaklardır. Görüldüğü gibi gerçek ve temsili argümanlar sayı ve tür bakımından aynıdır. Ayrıca altprogramdaki temsili argüman S bir dizi ismi olduğu için altprogramdaki dizinin boyutlarını belirten bir DIMENSION deyimi almıştır. Aynı şekilde ana programda S'nin yerini alan gerçek argüman GUNHAS'da bir dizi ismi olacağı için bu argümanın boyutlarını belirten bir DIMENSION deyimi ana programda da yer almıştır. Dizinin büyüklüğünü belirten gerçek argüman bir tamsayı sabiti (30) olarak verildiğinden, temsili argüman (N)'nin yerini alacaktır. Böylece aralarında ilişki kurulan ve dizi ismi olan argümanlar ana ve altprogramda aynı boyut büyüklüğüne sahip olmaktadırlar.

SUBROUTINE altprogramında görüldüğü gibi S ve N temsili argümanları altprograma veri sağlamakta ve program içinde de değerleri değişmemektedir. Yani bu iki argüman girdi argümanlarıdır. ENKUC ve L temsili argümanlarının değerleri altprogram içinde hesaplanıyor ve ana programa EKHAS ve NO gerçek argümanları adı altında götürülüyor. Dolayısiyle bu iki argüman çıktı argümanlarıdır. Böylece ana program ile altprogram arasında iletişimi sağlayan tüm argümanlar belirlenmiş olmaktadır.

Altprogramdaki DO döngüsünün işleyişi sona erdikten sonra kontrol RETURN deyimi aracılığı ile ana programda çağıran ifadesini izle-

yen WRITE deyimine geçmekte ve çıktı argümanlarının değerleri yazdırılmaktadır. Daha öncede belirtildiği gibi bu değerler ana programa temsili çıktı argümanlarının yerini alan EKHAS ve NO gerçek argümanları ile götürülmektedir. Çıktının yazdırılmasından sonra program işleyişi bitmektedir.

Örnek 2 : Stok Kontrol Problemlerinde Benzetim (Simülasyon) Tekniği ile İstemi Kestirme.

Stok kontrol problemlerinde istemi benzetim tekniği ile kestirmek geçmiş dönemlere ait verilere göre geliştirilmiş olan olasılık dağılımlarına dayalı olarak yapılır. Bunun için günlük istem miktarlarının nisbi frekansları (olasılıkları) toplamı 1.0 olan yığınsal olasılık dağılımına dönüştürülür ve yığınsal olasılık aralıklarına göre 00 ile 99 arasında rassal (tesadüfi) sayı aralıkları oluşturulur. Daha sonra rassal sayı üreten bir mekanizma ile bir rassal sayı üretilir. Rassal sayı hangi aralığa karşılık geliyorsa o aralığa giren günlük istem miktarı o günün istemi olarak alınır. Böyle üretilen günlük istem miktarları dinamik stok kontrol çalışmalarında yöneticilere büyük kolaylık sağlayacaktır.

Böyle bir çalışmada A işletmesine ait bilgiler aşağıdaki tabloda gösterildiği gibi olsun.

Günlük İstem (birim)	Olasılık	Yığınsal Olasılık	Rassal Sayı Aralığı
2	0.30	0.30	00 — — 29
3	0.20	0.50	30 — — 49
4	0.25	0.75	50 — — 74
5	0.15	0.90	75 — — 89
6	0.10	1.00	90 — — 99

Tabloya göre örneğin rassal sayı üretme mekanizmamız 68 rassal sayısını üretmiş ise bu rassal sayı 50 — — 74 aralığındadır ve buna karşılık gelen günlük istem 4 birimdir. Hemen belirtilmelidir ki rassal sayı üreten bir mekanizma SUBROUTINE altprogramı olarak çoğu bilgisayar sistemlerinde mevcuttur.

Yukarıdaki veri ve açıklamalara göre istemi kestirebilecek bir SUBROUTINE altprogramı ve onu kullanacak ana programın çağıran

ifadesi aşağıda verilecektir. Ancak daha önce kullanılacak değişkenler ve altprogramın akış şeması açıklanacaktır.

Değişkenler :

STOK : İstediğimiz altprogramın adı.

ISTEM : Günlük istem miktarı (birim olarak).

RASSAL : STOK altprogramı ile istemi kestirirken gereksinim duyduğumuz rassal sayıları üreten ikinci bir SUBROUTINE alt programı. Böyle bir programın bilgisayar sisteminde kullanıma hazır olduğu varsayılacaktır.

ID : RASSAL altprogramı ile sayı üretmek için veri olarak gerekli olan başlangıç değerini belirten değişken.

IY : RASSAL altprogramında üretilecek rassal bir tamsayı sabiti.

YFL : RASSAL altprogramında üretilecek (0 — 1) arasında bir gerçel rassal sayı.

IX : RASSAL altprogramında temsili argüman.

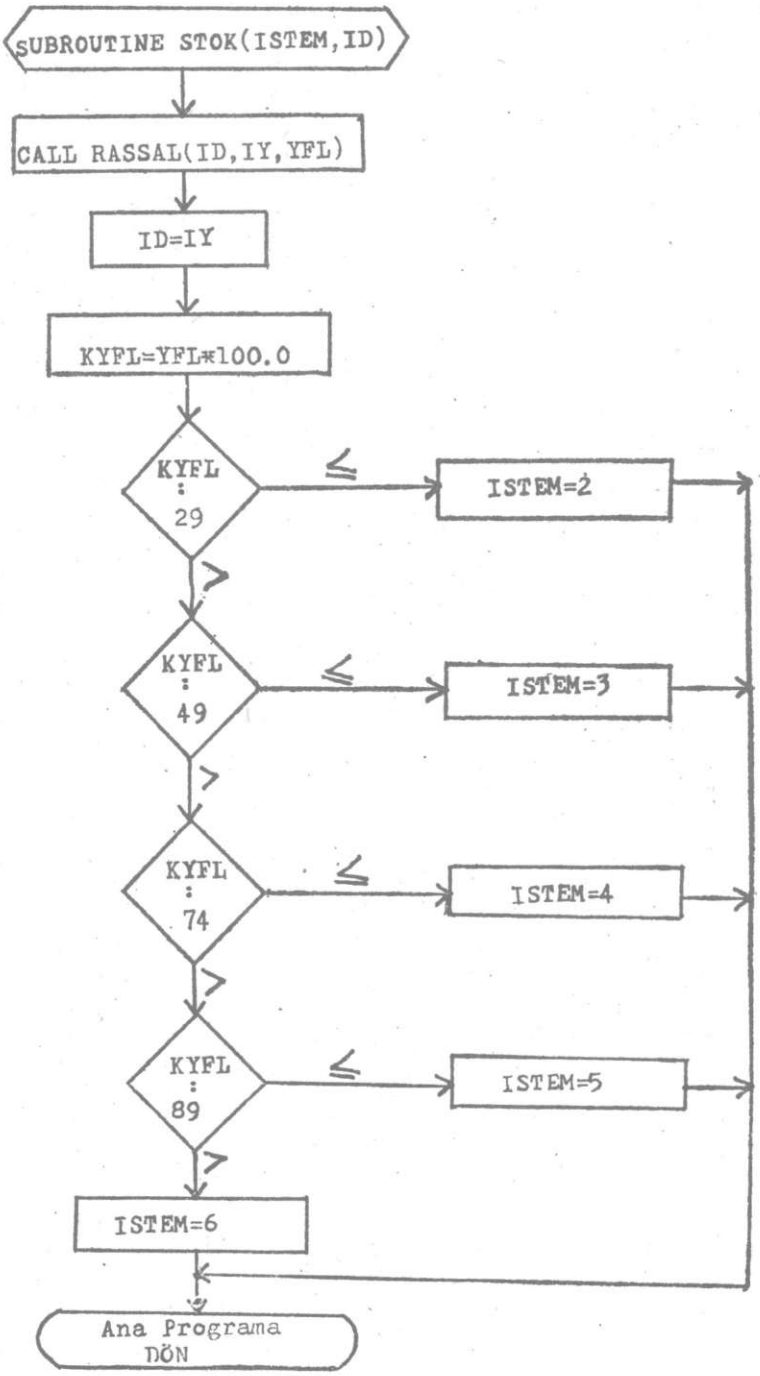
KYFL : YFL gerçel rassal sayısının (00 — 99) arasındaki tamsayı karşılığı.

Programda kullanılacak altprogramlar :

SUBROUTINE STOK (ISTEM, ID) : Burada ISTEM ve ID sırası ile çıktı ve girdi argümanlarıdırlar.

SUBROUTINE RASSAL (IX, IY, YFL) : Hazır olduğu varsayılan ve rassal sayı üreten altprogram.

STOK Altprogramı Akış Şeması :



Ana Programdaki Temel İfadeler :

```
      :
      ID=5647
      :
      :
      CALL STOK (ISTEM, ID)
      WRITE (5,6) ISTEM
      :
      :
      :
      STOP
      END
```

Altprogram :

```
      SUBROUTINE STOK (ISTEM, ID)
      CALL RASSAL (ID, IY, YFL)
      ID=IY
      KYFL=YFL★100.0
      IF (KYFL—29) 5,5,10
    5 ISTEM=2
      GO TO 99
   10 IF (KYFL—49) 15,15,20
   15 ISTEM=3
      GO TO 99
   20 IF (KYFL—74) 25,25,30
   25 ISTEM=4
      GO TO 99
   30 IF (KYFL—89) 35,35,40
   35 ISTEM=5
      GO TO 99
   40 ISTEM=6
   99 RETURN
      END
```

Rassal sayı üretme altprogramı :

```
      SUBROUTINE RASSAL (IX, IY, YFL)
      :
      IY=.....
      :
      YFL=.....
      :
      RETURN
      END
```

Programda görüldüğü gibi rassal sayı üretimi için gerekli olan başlangıç değeri ana programda ID=5647 olarak bildirilmektedir. Sonra günlük istem miktarını (ISTEM) belirlemek için STOK altprogramı CALL STOK (ISTEM, ID) olarak çağrılmaktadır. Çağıran ifadesinde gerçek argüman olarak ISTEM ve ID değişkenleri verilmektedir. Burada ISTEM çıktı ID ise girdi argümanlarıdır. Kontrol SUBROUTINE STOK (ISTEM, ID) altprogramına geçtiği zaman gerçek argümanlar aynı adı taşıyan ISTEM ve ID temsili argümanlarının yerini sırası ile alacaklardır. Görüldüğü gibi ISTEM ve ID değişkenleri hem gerçek hem de temsili argümanlar olarak kullanılmaktadır. Altprogram bağımsız olduğu için bu bir karışıklığa neden olmayacaktır.

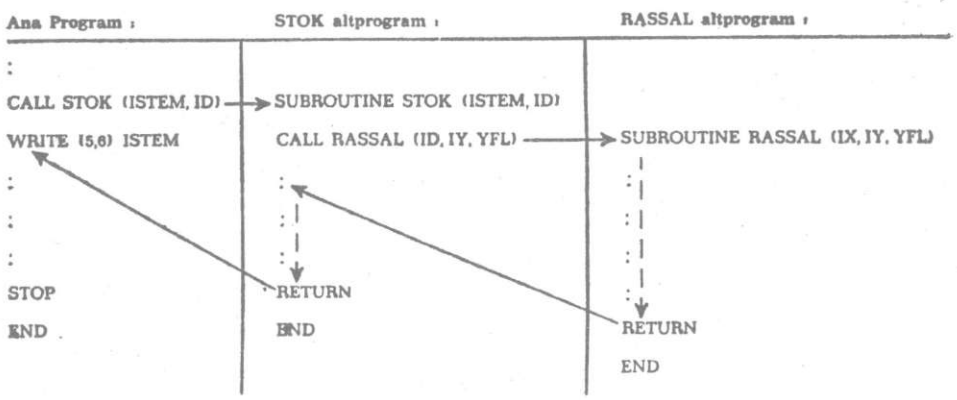
Bundan sonra SUBROUTINE STOK altprogramı içinde CALL RASSAL (ID, IY, YFL) ifadesi ile RASSAL altprogramı çağrılmaktadır. Böylece bir altprogram içinde bir başka altprogram çağrılmaktadır. Ancak bu kez ID gerçek bir girdi argümanı (değeri temel ana programda belirtilmiş ve daha sonra bir çağıran ifadesi ile STOK altprogramına gönderilmişti) olarak kullanılıyor ve RASSAL altprogramında, rassal sayı üretimi için gereksinim duyulan IX temsili argümanının yerini alacaktır. Çağıran ifadesindeki IY ile YFL gerçek, RASSAL altprogramındaki IY ile YFL ise temsili argümanlardır. Bu değişkenler (argümanlar) aynı gibi görünmekle beraber bilgisayara göre farklı değişkenler olarak işlem görecekler ve birinciler ikincilerin yerini alacaklardır.

RASSAL altprogramı konuyu karmaşıklaştırmamak için açık olarak yazılmamış olmasına karşın IX temsili argümanı yerine ID gerçek argümanını alıp rassal iki sayı üretmektedir. IY ve YFL temsili argümanları ile gösterilen bu sayılar çağıran programa aynı adı taşıyan gerçek argümanlar (IY, YFL) tarafından götürülmektedir. Bilindiği gibi çağıran program (ana program) bu kez SUBROUTINE STOK altprogramıdır. Bu aşamada kontrol artık ID=IY ifadesine geçmiştir. Yani RASSAL altprogramında üretilen IY değeri ID'ye verilmektedir. Bunun nedeni tekrar rassal sayı üretmek gerektiği zaman değeri bilinen farklı bir rassal sayı başlangıç değerini hazır bulundurmaktır. Böylece üretilen her rassal tamsayı bir sonraki aşamada üretilecek olan ikinci rassal tamsayı için bir başlangıç değeri oluşturmaktadır.

Diğer yandan RASSAL altprogramı tarafından üretilmiş olan ikinci sayı YFL hazır bulunan altprogram gereği (0 — 1) arasında gerçel bir sayıdır. Bu sayıyı iki basamaklı bir tamsayı sabitine dönüştürmek için 100.0 ile çarpıp sonucu bir tamsayı değişkeni olan KYFL'ye vermek yeterli olacaktır. Böylece KYFL=YFL★100.0 ifadesi sonucu olan

KYFL iki basamaklı yani (00 — 99) arasında rassal bir tamsayı olacaktır. Bu sayıyı yığınsal olasılık dağılımına dayalı olarak belirlenen rassal sayı aralıkları ile karşılaştırır ve uyduğu aralığa göre günlük istem miktarını (ISTEM) belirleriz. Bir ISTEM değeri belirlendikten sonra kontrol 99 RETURN deyimine geçmekte ve buradan da ilk ana programa götürülmektedir. Bu aşamada bilgisayar işlem kontrolü WRITE (5,6) ISTEM ifadesine geçecektir.

Özet olarak, ISTEM değişkeninin bir tek değerini üretmek için bir rassal sayı başlangıç alınıp STOK altprogramına gidilmekte ve buradan da RASSAL altprogramına rassal bir sayı üretimi için gidiliyor. Rassal sayı burada üretildikten sonra tekrar STOK altprogramına bırakılan noktada dönülüyor ve ISTEM değişkeninin değeri karşılaştırmalarla belirleniyor. Bundan sonra kontrol tekrar ana programa bırakılan noktaya gönderiliyor. Şekil olarak program akışı aşağıdaki biçimde belirtilebilir.



3.5 Altprogramlarla İlgili Bazı Deyimler

3.5.1 COMMON Deyimi

Daha öncede belirtildiği gibi ana program ile altprogramlar arasındaki iletişim argümanlar aracılığı ile olmaktadır. Ana program ve altprogramlar bir birinden ayrı programlar oldukları için her programda bulunan değişken aynı adı taşısa bile bellekte ayrı bir yerde depolanır. Ancak COMMON deyimi kullanılmakla ana programdaki gerçek argümanlar ile altprogramdaki temsili argümanları aynı yerlerde depolamak ve böylece argüman sayısını azaltmak mümkün olmaktadır. Örneğin aşağıdaki ana ve altprogramdaki argümanları ele alalım.

Ana program :

```
:
CALL MAL (S,F,L,M)
:
:
```

Altprogram :

```
SUBROUTINE MAL (A,B,I,J)
:
:
```

Bilindiği gibi ana programda MAL altprogramı çağrıldığı zaman S,F,L,M gerçek argümanları A,B,I,J temsil argümanlarının yerini alacaktır. Fakat COMMON deyimi kullanılırsa yukarıdaki ifadeler şöyle yazılabilir.

Ana program :

```
COMMON S,F,L,
CALL MAL (M)
:
:
```

Altprogram :

```
SUBROUTINE MAL (J)
COMMON A,B,I
:
:
```

Böylece kullanılan COMMON deyimi ile S ile A, F ile B ve L ile I bellekte aynı yeri (ortak kullanım alanı) kullanacaklardır. Dolayısiyle aynı kullanım alanı içindeki değişkenler aynı değerleri alacaklardır. Yani S ile A, F ile B ve L ile I aynı değere sahip olacaklardır. Bu nedenle bu değişkenler arasında iletişimi sağlamak için bunları argüman olarak belirtmeye gerek yoktur. Çünkü iletişim COMMON deyimi ile sağlanmıştır. Böylece tek gerçek argüman olarak M ve karşılığı temsili argüman olan J kalmıştır ve argüman sayısı da azaltılmış olmaktadır. İstenilirse M ve J argümanları da her iki programda aynı ortak kullanım alanına alınabilir. Bu durumda altprogramın argümanı olmayacaktır. Yani, çağıran ifadesi CALL MAL ve altprogramın ilk ifadesi de SUBROUTINE MAL biçimini alacaktır. Bunlar ise geçerli ifa-

delerdir. Ancak FUNCTION altprogramlarında çoğu kez en az bir argüman bulunma zorunlu olmaktadır.

Diğer yandan ortak kullanım alanına alınan değişkenler (argümanlar) eğer dizinli ise COMMON deyiminde bunun belirtilmesi ile boyut işlemi de yapılmış olur. Örneğin, S ve A değişkenleri (10×15) boyutlu dizinli değişkenler ise,

Ana program :	Altprogram :
COMMON S (10,15), F,L	COMMON A (10,15), B,I

biçiminde yazılabilir. Kuşkusuz COMMON deyiminde boyut belirtilmediği durumlarda her iki programda DIMENSION deyimi kullanılacaktır. Ancak hem COMMON deyimi ile hem de DIMENSION deyimi ile aynı değişkenlerin boyutlarını belirtmek hatalı olur.

Şimdiye kadar yapılan açıklamaların da belirttiği gibi ana ve altprogramlar arasında iletişimi ve değişkenlerin bellek sisteminde aynı yeri ortak kullanmasını sağlayan COMMON deyimi ile biri ana programda diğeri (diğerleri) altprogramda olan değişkenler aynı ortak kullanım alanına alınmaktadır.

Bazı durumlarda ise eldeki program oldukça büyük, kullanılacak bilgisayar bellek sisteminin küçük olması nedeni ile programın gerektirdiği bellek alanını küçültmek için EQUIVALENCE deyimi kullanılır. Bu deyim ile ana programdaki ya da altprogramlardaki aynı türden iki ya da daha fazla sayıda değişkenin bellekte aynı yeri paylaşmaları sağlanır. Örneğin;

```
EQUIVALENCE (L,M,N),(R,C)
```

biçimindeki bir ifade ile aynı programdaki L,M ve N tamsayı değişkenleri bir depolanma yerini paylaşacak, R ve C gerçel değişkenleri ise bir başka yeri paylaşacaklardır. Örnekte görüldüğü gibi aynı programdaki değişkenleri ortak bir kullanım alanına alan EQUIVALENCE deyiminde ortak kullanım alanına alınan aynı türdeki değişkenler bir birinden virgül ile ayrılıp ayraç içine alınmaktadır. Ayrıca parantez içindeki gruplarda virgül ile bir birinden ayrılmışlardır. Bu biçim deyimin genel kullanım kalıbıdır. Değişkenler dizinli değişkenlerde olabilir. Kuşkusuz, COMMON deyimi ile EQUIVALENCE deyimleri birlikte kullanılabilir. Ancak böyle durumlarda COMMON deyiminde yer alan değişkenlerden 1'den fazlasının EQUIVALENCE deyimini oluşturan aynı grubun içinde olmaması gerekir. Örneğin;

```
COMMON A,B,C
EQUIVALENCE (A,B,Z)
```

ifadelerinde COMMON deyimine göre bellek sisteminde bir birini izleyen üç depolama bölgesi A,B ve C değişkenleri için ayrılmıştır. Ancak EQUIVALENCE deyimine göre de A,B, ve Z değişkenlerininde aynı depolama alanını paylaşmaları istenmiştir. Böyle bir durumda, A ve B değişkenleri COMMON deyimine göre iki ayrı bölgede fakat EQUIVALENCE deyimine göre de aynı bölgeyi paylaşmaları istendiği için COMMON ve EQUIVALENCE deyimlerinin ifadeleri bir biri ile çelişkilidir. Dolayısiyle yukarıdaki ifadeler hatalı yazılmışlardır.

3.5.2 EXTERNAL Deyimi

Bir altprogramı çağıran ifadede yer alan gerçek argümanlar altprogram adları da olabilirler. Başka bir ifade ile bir altprogram adı başka bir altprogramı çağıran ifadenin gerçek argümanı olarak kullanılabilir. Ancak bunu yapabilmek için çağıran ifadesinin yer aldığı ana programda altprogram adının bir EXTERNAL deyimi ile belirtilmesi gerekir. Bunun için aşağıdaki genel biçim izlenmelidir.

EXTERNAL $a_1,a_2,....,a_n$

Burada $a_1,....,a_n$ çağıran ifadesinde yer alan ve altprogramlara aktarılan altprogram adlarını temsil etmektedir. Örneğin, HIZMET adındaki bir SUBROUTINE altprogramını çağırırken gerçek argümanlar arasında SIRA adlı bir başka altprogramın (FUNCTION ya da SUBROUTINE) adı bulunsun. Yani, SIRA altprogramının adı bir gerçek argüman olarak HIZMET altprogramına iletilmek istensin. Bu durumda,

Ana program :	Altprogram :
EXTERNAL SIRA	SUBROUTINE HIZMET (X,Y,Z)
:	:
:	:
CALL HIZMET (A,B, SIRA)	:
:	:

biçiminde yazılır. Kuşkusuz, birden fazla altprogram adı EXTERNAL deyimi aracılığı ile bir başka altprograma gerçek argüman olarak aktarılabilir.

3.5.3 ENTRY Deyimi

Bu aşamaya kadar verilen açıklamalarında belirttiği gibi, bir FUNCTION ya da SUBROUTINE altprogramı çağrıldığı zaman alt

programa giriş bu sözcüklerin bulunduğu ilk ifadeden başlar ve RETURN deyimi ile kontrol çağıran ana programa döner. Fakat bazı FORTRAN IV derleyicileri ENTRY deyimi ile bir altprograma istenilen noktalardan girme olanağı sağlar. ENTRY deyimi işlenemez bir deyim olup aşağıdaki genel biçimde kullanılır.

ENTRY f (d_1,d_2,.....,d_n)

Burada;

f : Altprograma giriş noktasının adını,

d_1,...,d_n : Temsili argümanları göstermektedir.

Altprogram içerisinde böylece belirlenen noktadan girişi yapacak ve altprogramı çağıracak ifade ise CALL deyimi kullanılarak aşağıdaki gibi yazılır.

CALL f (a_1,a_2,.....,a_n)

Burada;

f : Altprogramda ENTRY deyimi ile belirtilen giriş noktasının adını,

a_1,...,a_n : Gerçek argümanları temsil etmektedir.

ENTRY noktaları bir altprogram olarak değerlendirileceği için ENTRY deyimi argümanları içerisinde bulundukları FUNCTION ya da SUBROUTINE altprogramlarının argümanları ile uyuşum içinde bulunmak zorunda değildirler. Örneğin;

```
Ana program :                    Altprogram :
:                                SUBROUTINE GELIR (X,Y,W,Z)
CALL GELIR (A,B,C,D)             :
:                                :
CALL YAD (L,G)                   ENTRY YAD (I,D)
:                                :
CALL AR (T,F)                    :
:                                ENTRY AR (P,R)
:                                :
STOP                             :
END                              RETURN
                                 END
```

biçiminde genel hatları ile belirtilen bir programı düşünelim. Programda GELIR adını taşıyan bir SUBROUTINE altprogramı ve bu altprog-

ramda YAD ve AR adında iki giriş noktası bulunmaktadır. Ana programda CALL GELIR (A,B,C,D) çağıran ifadesi ile altprogram çağrıldığı zaman buradaki gerçek argümanlar sırası ile GELIR altprogramının X,Y,W,Z temsili argümanlarının yerini alacak ve altprogramın belirttiği işlemler yapılmaya başlanacaktır. İşlem sırasında ENTRY YAD (I,D) ve ENTRY AR (P,R) giriş noktaları için her hangi bir işlem yapılmayacaktır. Çünkü ENTRY deyimi işlenemez bir deyimdir. Böylece altprogramın işlemi RETURN deyimine ulaşınca kontrol CALL GELIR (A,B,C,D) ifadesini izleyen ifadeye geçer.

İşlem sırası CALL YAD (L,G) çağıran ifadesine geldiği zaman kontrol tekrar altprograma dönecektir. Ancak bu kez işlem ENTRY YAD (I,D) noktasını izleyen ifadeden başlayacak ve RETURN deyimine kadar devam edecektir. Yine kontrol ana programa ve CALL YAD (L,G) ifadesini izleyen deyime geçecek ve işleme devam edecektir. Aynı şekilde işlem sırası CALL AR (T,F) ifadesine geldiği zaman kontrol, altprogramda ENTRY AR (P,R) ifadesini izleyen ifadeden T,P'nin yerini ve F argümanıda R'nin yerini alarak işlem devam edecektir. Sonra yine kontrol RETURN deyimi ile ana programa geçecek ve istenilen işlemler ana program bitinceye kadar devam edecektir.

IV.

PROGRAMLAMA ÖRNEKLERİ

Örnek 1 : Bir Tablonun Sütun ve Sıra Elemanlarının Toplanması

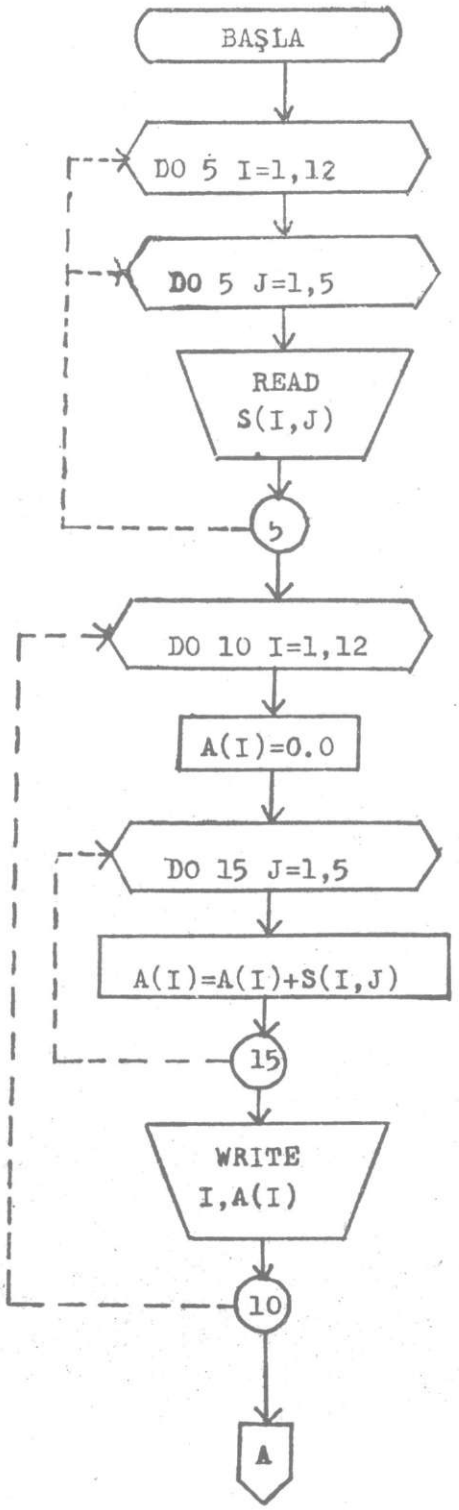
Aşağıdaki 12 sıralı ve 5 sütunlu $S_{i,j}$ tablosunda i sıraları aylara göre, j sütunları ise yıllara göre satış miktarlarını göstermektedir.

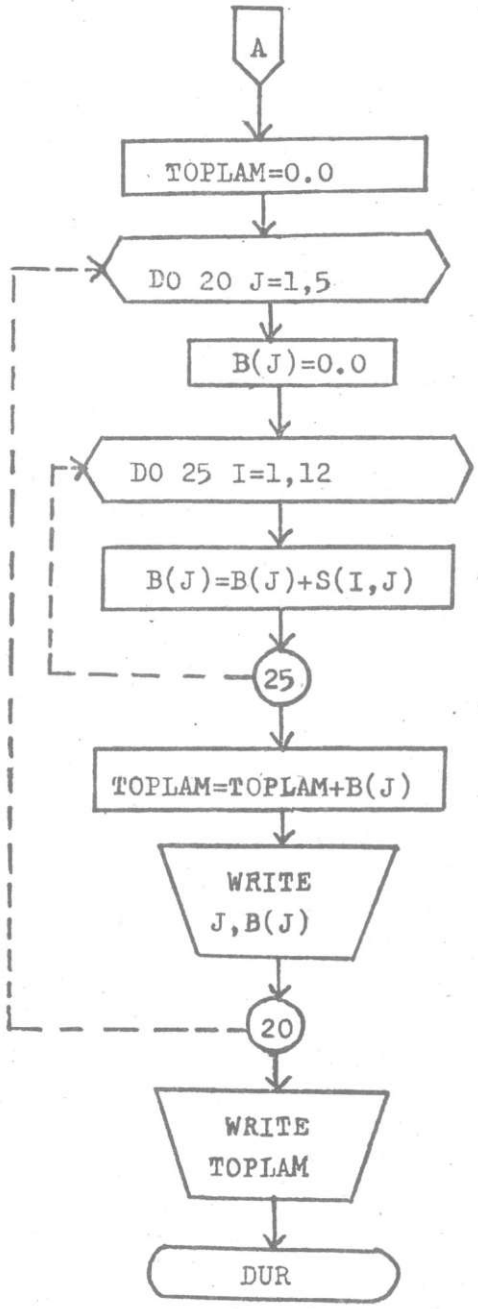
	1977	1978	1979	1980	1981
Ocak	12	25	18	13	17
Şubat	7	6	20	5	15
Mart	5	12	45	16	4
Nisan	6	14	20	19	41
Mayıs	7	13	45	13	34
Haziran	9	11	26	6	12
Temmuz	14	16	32	19	24
Ağustos	30	25	40	33	24
Eylül	15	15	23	6	17
Ekim	24	16	4	10	23
Kasım	43	37	10	8	27
Aralık	55	25	15	7	13

Aylara ve yıllara göre satış miktarlarının toplamını ayrı ayrı hesaplayacak bir program hazırlayalım.

Değişkenler :

S (I,J) : Satış miktarı.
I : Ayları gösteren sıra dizini, I=1,2,.....,12.
J : Yılları gösteren sütun dizini, J=1,2,...,5.
A (I) : I'nci ayda tüm yıllarda yapılan satış toplamı.


BAŞLA
DO 5 I=1,12
DO 5 J=1,5
READ
S(I,J)
5
DO 10 I=1,12
A(I)=0.0
DO 15 J=1,5
A(I)=A(I)+S(I,J)
15
WRITE
I,A(I)
10
A


A
TOPLAM=0.0
DO 20 J=1,5
B(J)=0.0
DO 25 I=1,12
B(J)=B(J)+S(I,J)
25
TOPLAM=TOPLAM+B(J)
WRITE
J,B(J)
20
WRITE
TOPLAM
DUR

Örneğin, birinci aydaki (Ocak) satış toplamı;

$$A(1) = \sum_{j=1}^{5} S_{1j} = 85\text{'dir.}$$

B (J) : J'nci yılda yapılan satışlar toplamı. Örneğin birinci yıldaki (1977) satış toplamı;

$$B(1) = \sum_{i=1}^{5} S_{i1} = 227$$

TOPLAM : Tablodaki tüm satışlar toplamı.

Program :

```
      DIMENSION S(12,5), A(12), B(5)
      DO 5 I=1,12
      READ (5,7) (S(I,J), J=1,5)
    5 CONTINUE
    7 FORMAT (5F4.0)
C   SIRA TOPLAMININ HESAPLANMASI
      DO 10 I=1,12
      A(I)=0.0
      DO 15 J=1,5
      A(I)=A(I)+S(I,J)
   15 CONTINUE
      WRITE (6,8) I,A(I)
   10 CONTINUE
    8 FORMAT (5X, 'SIRA', 3X, I2, 3X, 'TOPLAM=', F6.0)
C   SUTUN TOPLAMININ HESAP.
      TOPLAM=0.0
      DO 20 J=1,5
      B(J)=0.0
      DO 25 I=1,12
      B(J)=B(J)+S(I,J)
   25 CONTINUE
      TOPLAM=TOPLAM+B(J)
      WRITE (6,9) J,B(J)
   20 CONTINUE
    9 FORMAT (5X, 'SUTUN', 2X, I2, 3X, 'TOPLAM=', F6.0)
C   TUM TABLO TOPLAMINI YAZDIRMA
      WRITE (6,11) TOPLAM
```

```
   11 FORMAT (5X, 'TUM TABLO TOPLAMI=', F8.0)
      STOP
      END
```

Bu programa göre tablodaki veriler sıra numarasına göre her sıra bir karta delinerek 12 kartta okutulacaktır. Örneğin, birinci ve ikinci veri kartları aşağıdaki gibi olacaktır :

```
12345678... ... ... ... ... ... ... ... ... ... ... ... 80 (sütun)
1. kart     bb12bb25bb18bb13bb17
2. kart     bbb7bbb6bb20bbb5bb15
```

Program çıktısı ise aşağıdaki gibi yazılacaktır :

```
SIRA       1     TOPLAM=    85.
SIRA       2     TOPLAM=    53.
SIRA       3     TOPLAM=    82.
SIRA       4     TOPLAM=   100.
SIRA       5     TOPLAM=   112.
SIRA       6     TOPLAM=    64.
SIRA       7     TOPLAM=   105.
SIRA       8     TOPLAM=   152.
SIRA       9     TOPLAM=    76.
SIRA      10     TOPLAM=    77.
SIRA      11     TOPLAM=   125.
SIRA      12     TOPLAM=   115.
SUTUN      1     TOPLAM=   227.
SUTUN      2     TOPLAM=   215.
SUTUN      3     TOPLAM=   298.
SUTUN      4     TOPLAM=   115.
SUTUN      5     TOPLAM=   251.
TUM TABLO  TOPLAMI=     1146.
```

Örnek 2 : İki Matris Çarpımının Hesaplanması

A_{ij} ve B_{mk} gibi iki boyutlu iki matrisin çarpılıp bulunacak C matrisinin hesaplanması istenmektedir. Ancak hemen belirtilmelidir ki A_{ij} ve B_{mk} matrislerinin çarpılabilmesi için A matrisinin sütun sayısı ile B matrisinin sıra sayısının eşit olması gereklidir. Yani, j=m olmalıdır.

Çarpım sonucu elde edilecek C matrisi ise C_{ik} biçiminde iki boyutlu bir matris olacaktır. Çarpma işlemi aşağıdaki yönteme göre yapılacaktır.

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1j} \\ A_{21} & A_{22} & \dots & A_{2j} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ A_{i1} & A_{i2} & \dots & A_{ij} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1k} \\ B_{21} & B_{22} & \dots & B_{2k} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mk} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1k} \\ C_{21} & C_{22} & \dots & C_{2k} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ C_{i1} & C_{i2} & \dots & C_{ik} \end{bmatrix}$$

Elde edilen C matrisinin her elemanı, A matrisinin sırası ile buna karşılık gelen B matrisinin sütununu oluşturan elemanların ayrı ayrı çarpılması ve elde edilen sonuçların toplanıp C matrisinin karşılık gelen elemanı olarak yazılması ile elde edilir. Yani;

$$\begin{aligned} C_{11} &= A_{11}\ B_{11} + A_{12}\ B_{21} + A_{13}\ B_{31} + \dots\dots\dots + A_{1j}\ B_{m1} \\ C_{12} &= A_{11}\ B_{12} + B_{12}\ B_{22} + A_{13} B_{32} + \dots\dots\dots + A_{1j}\ B_{m2} \\ &\vdots \quad \vdots \qquad \vdots \qquad \vdots \qquad\qquad \vdots \end{aligned}$$

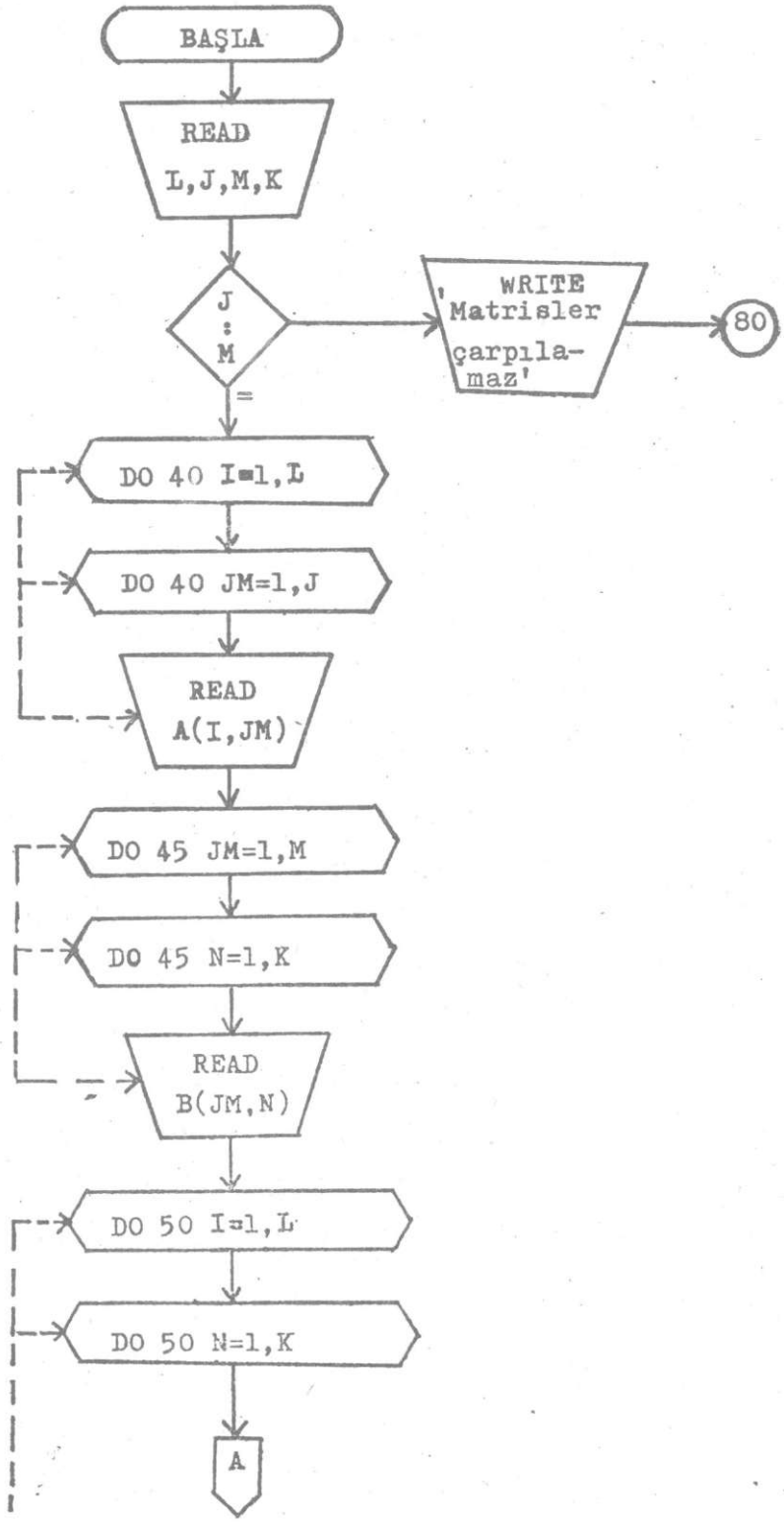
gibi.

Örneğin;

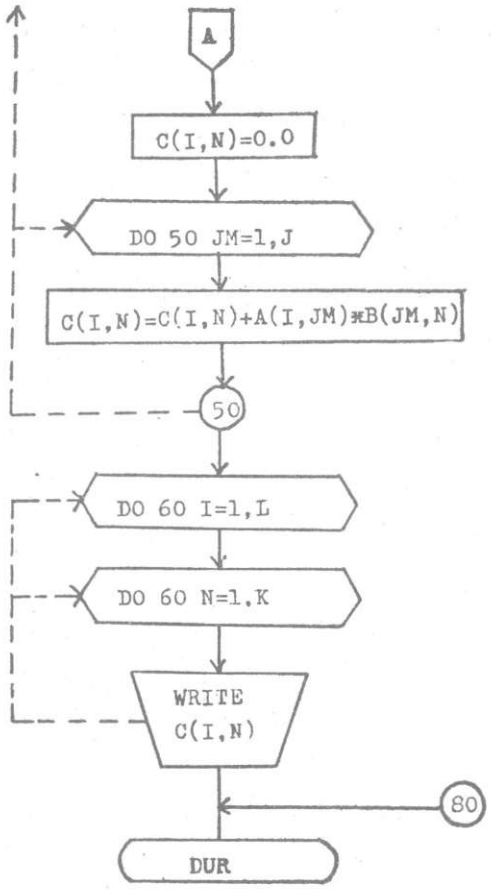
$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} X \begin{bmatrix} 3 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} (1\times3+2\times0) & (1\times0+2\times1) & (1\times1+2\times1) \\ (1\times3+0\times0) & (1\times0+0\times1) & (1\times1+0\times1) \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 2 & 3 \\ 3 & 0 & 1 \end{bmatrix}$$ bulunur.

Yukarıda belirtilen yönteme göre programın akış şeması aşağıdaki gibi çizilebilir.


BAŞLA
READ
L,J,M,K
J
:
M
=
WRITE
'Matrisler
çarpıla-
maz'
80
DO 40 I=1,L
DO 40 JM=1,J
READ
A(I,JM)
DO 45 JM=1,M
DO 45 N=1,K
READ
B(JM,N)
DO 50 I=1,L
DO 50 N=1,K
A

Akış - Şeması



Yukarıdaki akış şemasına göre birinci matrisin sütun sayısı J ile ikinci matrisin sıra sayısı M eşit değilse, iki matris çarpılamaz. Dolayısiyle program işlemeyecek ve hemen duracaktır. Eğer matrisler çarpım için uygun ise çarpım işlemi için iç içe yuvalanmış döngü alanı içine kontrol aktarılmaktadır. Bu iç içe üç döngünün işleyişi L=2, K=3 ve M=J=2 örneği için, yani A(2×2) ve B(2×3) matrisleri için, aşağıdaki gibi açıklanabilir (DO 50 I=1,L nin bir tekrarı için).

İşlem Sayısı	I	N	JM	C(I, N) = C(I, N) + A(I,JM) ★ B(JM,N)
1	1	1	1	C(1, 1) = C(1, 1) + A(1, 1) ★ B(1, 1)
2	1	1	2	C(1, 1) = C(1, 1) + A(1, 2) ★ B(2, 1)
3	1	1	3	C(1, 1) = C(1, 1) + A(1, 3) ★ B(3, 1)
4	1	2	1	C(1, 2) = C(1, 2) + A(1, 1) ★ B(1, 2)
5	1	2	2	C(1, 2) = C(1, 2) + A(1, 2) ★ B(2, 2)
6	1	2	3	C(1, 2) = C(1, 2) + A(1, 3) ★ B(3, 2)
7	1	3	1	C(1, 3) = C(1, 3) + A(1, 1) ★ B(1, 3)
8	1	3	2	C(1, 3) = C(1, 3) + A(1, 2) ★ B(2, 3)
9	1	3	3	C(1, 3) = C(1, 3) + A(1, 3) ★ B(3, 3)

Görüldüğü gibi I=1 için işlem bittiğinde C matrisinin ilk sırasının elemanları C(1,1), C(1,2) ve C(1,3) hesaplanmış olmaktadır. I=2 için son kez işlem yapıldığında ise C(2,1), C(2,2) ve C(2,3)'de yukarıdaki gibi hesaplanmış olacaktır. Böylece C(2×3) gibi iki boyutlu bir matris elde edilecektir.

A (5×10) ve B (10×15) boyutlarında iki matris olarak düşünülürse, yukarıda verilen akış şemasına göre bir program şöyle yazılabilir. Okutma ve yazdırma işlemleri için normal ve kapalı DO döngüleri birlikte kullanılmıştır.

```
C  IKI MATRIS CARPIMININ BULUNMASI
C
      DIMENSION A(5,10), B(10,15), C(5,15)
      READ (5,6) L,J,M,K
    6 FORMAT (4I3)
      IF (J-M) 30,35,30
   35 DO 40 I=1,L
   40 READ (5,7) (A(I,JM), JM=1,J)
      DO 45 JM=1,M
   45 READ (5,4) (B(JM,N), N=1,K)
    4 FORMAT (15F4.0)
    7 FORMAT (10F4.0)
      DO 50 I=1,L
      DO 50 N=1,K
      C(I,N) =0.0
```

```
      DO 50 JM=1,J
      C(I,N)=C(I,N)+A(I,JM)★B(JM,N)
50    CONTINUE
      DO 60 I=1,L
      WRITE (6,8) (C(I,N), N=1,K)
60    CONTINUE
      GO TO 80
30    WRITE (6,9)
 8    FORMAT (3X, 15(F6.0, 3X) )
 9    FORMAT (3X, 'BU MATRISLER CARPILAMAZ')
80    STOP
      END
```

Verilen programa göre, A matrisinin her sırasında yer alan 10 elemanı 7 FORMAT'a göre 1 karta delinecek ve böylece bu matrisin verilerini okutmak için 5 kart kullanılacaktır. 4 FORMAT'a göre okutulan B matrisi için de 10 kart gerekecektir. Yani bu matrisin 15 elemanlı her sırası için her biri 15 eleman okutan 10 kart kullanılacaktır. Böylece toplam 15 veri kartı kullanılacaktır. Kuşkusuz FORMAT'lara göre her eleman kartlarda 4 sütunluk bir alana delinirken alanın sağına delinecek ve boş sütunlar olursa solda bırakılacaktır.

Örnek 3 : Bir Dizinin Ortalama, Varyans ve Standart Sapmasının Hesaplanması

Bir S_n dizisinde;

$$\text{Aritmetik Ortalama (ARORT)} = \left(\sum_{i=1}^{n} S_i\right)/n \qquad (i=1,2,.....,n)$$

$$\text{Varyans (VARYN)} = \frac{\sum_{i=1}^{n} (S_i - \text{ARORT})^2}{n-1}$$

$$\text{Standart Sapma (SAPMA)} = \sqrt{\text{VARYN}}$$

Bu formüllere göre yazılacak programda altprogramlar da kullanabiliriz. Formülleri kısaltmak için yeni değişkenler oluşturalım :

$$\text{TOPLAM} = \sum_{i=1}^{n} S_i$$

$$\text{FARKTO} = \sum_{i=1}^{n} (S_i - \text{ARORT})^2$$

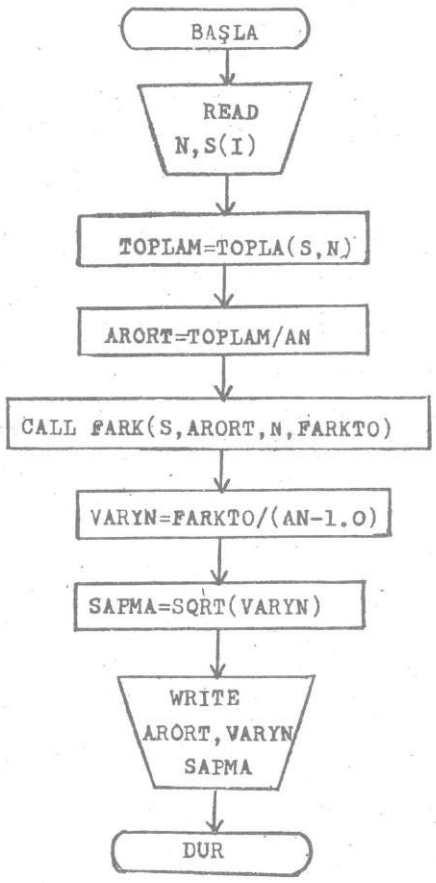
AN=N

VARYN=FARKTO/(AN−1.0)

TOPLA : TOPLAM değerini hesaplayacak FUNCTION altprogramı,

FARK : FARKTO değerini bulacak SUBROUTINE altprogramı adı.

Ana Program Akış Şeması :



Ana Program :

```
C  ORTALAMA VARYANS VE STAND. SAPMANIN
C  HESAPLANMASI
       DIMENSION S(100)
       READ (5,1) N, (S(I), I=1,N)
     1 FORMAT (I4,/, (10F8.2) )
```

```
C  FONKSIYON CAGIRMA
       TOPLAM=TOPLA (S,N)
       AN=N
       ARORT=TOPLAM/AN
C  ALTRUTIN CAGIRMA
       CALL FARK (S, ARORT, N, FARKTO)
       VARYN=FARKTO/(AN-1.0)
       SAPMA=SQRT (VARYN)
       WRITE (6,2) ARORT, VARYN, SAPMA
     2 FORMAT (5X, 'ORTALAMA=', F8.2/
      ★5X, 'VARYANS=', F8.2,/,
      ★5X, 'STANDART SAPMA=', F5.2)
       STOP
       END
C  FONKSIYON ALTPROGRAMI
       FUNCTION TOPLA (A,J)
       DIMENSION (A,J)
       TOPLA=0.0
       DO 1 I=1,J
     1 TOPLA=TOPLA+A(I)
       RETURN
       END
C  ALTRUTIN ALTPROGRAMI
       SUBROUTINE FARK (B,C,J,F)
       DIMENSION B(J)
       F=0.0
       DO 2 I=1,J
     2 F=F+(B(I)-C)★★2
       RETURN
       END
```

Örnek 4 : Aylık Ücret Bordrosunun Hazırlanması

İstihdam ettiği memurlarına personel yasası gereğince ücret ödeyen bir kuruluş için aylık ücret bordrosunu aşağıdaki yönteme göre hazırlayalım.

1. Personelin derece ve kademesine göre gösterge belirlendikten sonra maaş tutarı,

Maaş tutarı = Katsayı × Gösterge

biçiminde hesaplanır.

2. Maaş tutarına göre kurumca verilen % 18 emeklilik keseneği ile % 10 emekli keseneği hesaplanır.

3. Memurun medeni durumuna göre aile yardımı belirlenir. Eş çalışmıyorsa aile yardımı 600 liradır. Çalışıyorsa yardım yoktur. Eşi çalışan memur kadın ise çocuk yardımı yoktur. Çünkü erkek eş çocuk yardımı almaktadır. Daha sonra çocuk sayısı ve çocukların öğrenim düzeyine göre çocuk yardımı hesaplanmaktadır. Aile ve çocuk yardımları toplam aile yardımı olarak adlandırılacaktır.

4. Daha sonra gelir vergisi matrahı hesaplanmaktadır :

Gelir vergisi matrahı = Maaş tutarı + Yan ödeme — % 10 Emekli keseneği — Aile indirimi — Borçlanmalar.

5. Gelir vergisi = Gelir vergisi matrahı × 0.39 olarak hesaplanmaktadır. Yani gelir vergisi matrahından % 39 gelir vergisi alınmaktadır.

6. Mali denge vergisi = Gelir vergisi matrahı × 0.02 biçiminde hesaplanmaktadır.

7. Damga resmi = (Maaş tutarı + Yakacak yardımı + yan ödeme) × 0.004

olarak hesaplanmaktadır.

8. Yukarıdaki hesaplanmalara dayalı olarak,

Genel Toplam = Maaş tutarı + % 18 Emeklilik keseneği + Yakacak yardımı + Toplam aile yardımı + Yan ödeme

Kesinti Toplamı = % 18 Emeklilik keseneği + % 10 Emeklilik keseneği + Gelir vergisi + Mali denge vergisi + Damga resmi + Borçlanmalar

Net Ödenecek = Genel Toplam — Kesinti Toplamı

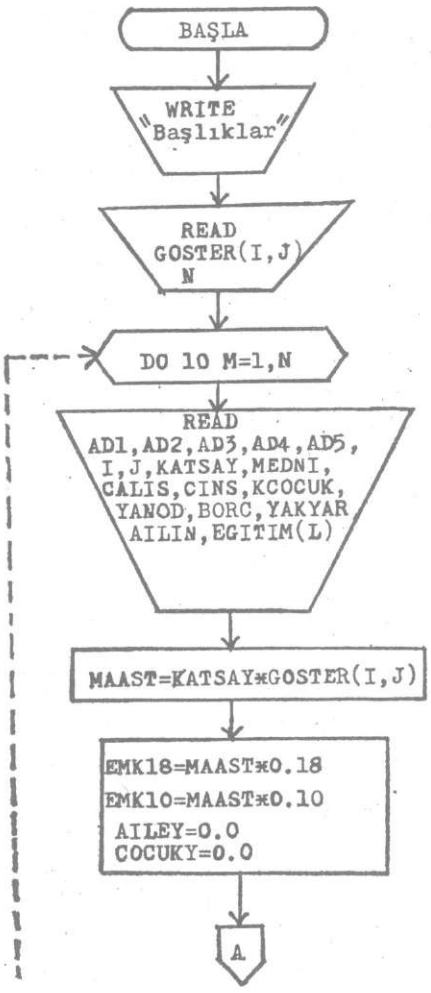
biçiminde hesaplanacaktır. Son olarak bordroda istenilen bilgiler yazılacaktır.

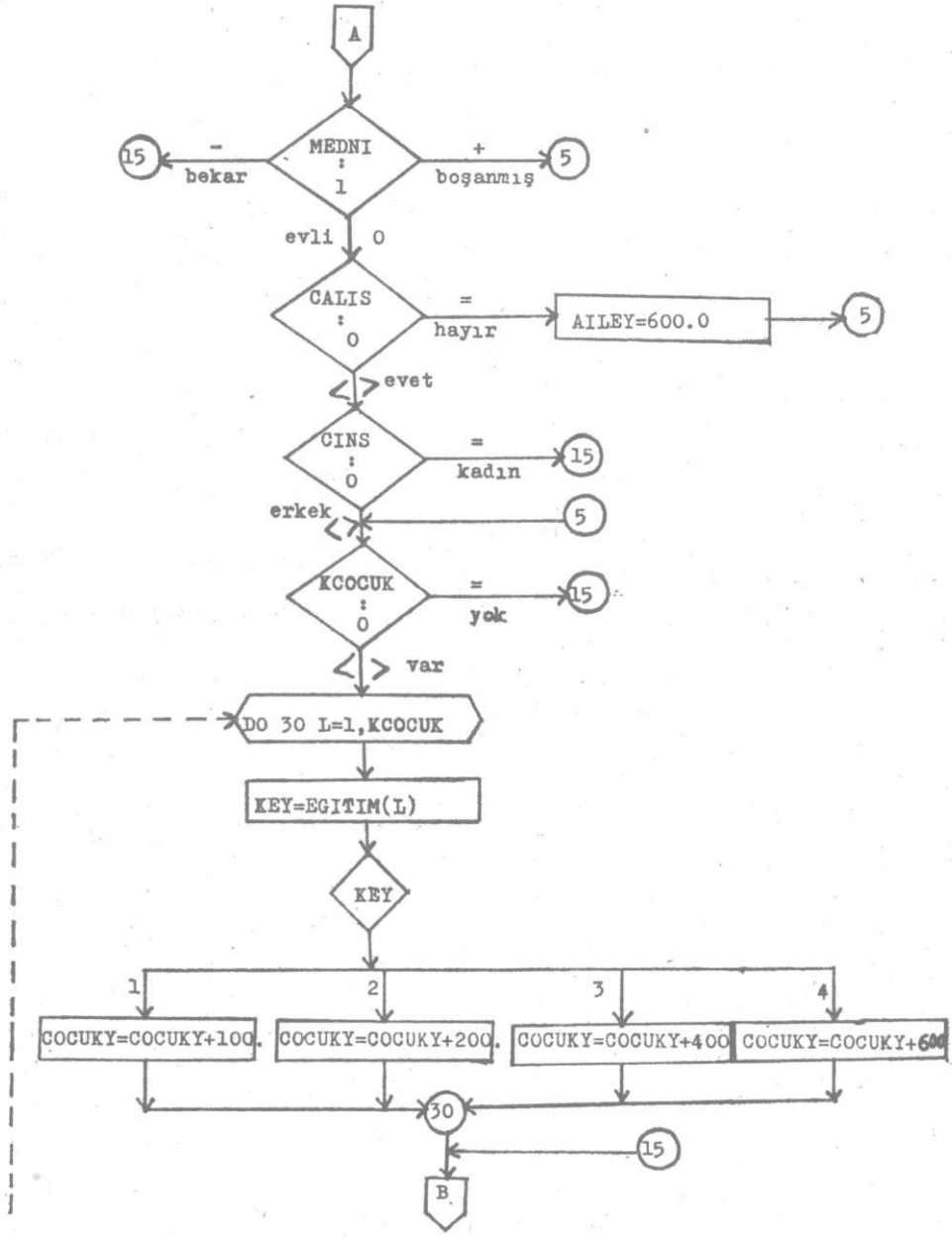
Değişkenler :

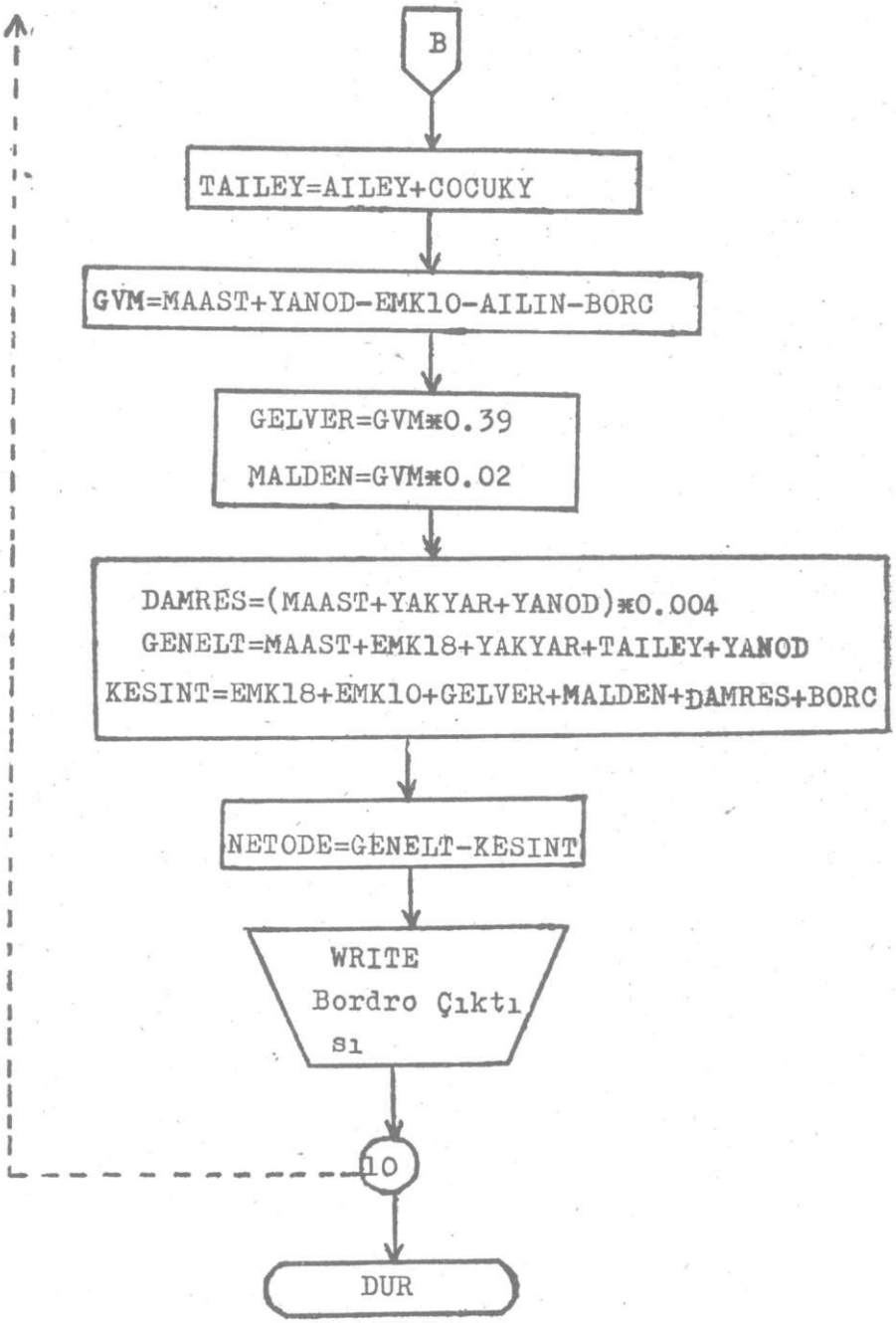
N	: Bordrodaki memur sayısı.
I	: Memurun bulunduğu derece, I=1,2,....,15.
J	: Memurun bulunduğu kademe, J=1,2,....,9.
GOSTER (I,J)	: I'nci derecede ve J'nci kademede aylık gösterge.
ADI,...,AD5	: Memurların adı, soyadı ve sicil numarası.
MAAST	: Maaş tutarı.
KATSAY	: Katsayı
EMK18	: Kurumca verilen % 18 emekli keseneği.
EMK10	: % 10 emekli keseneği.
AILEY	: Aile yardımı.
COCUKY	: Çocuk yardımı. Okula gitmeyen için 100, ilkokula giden için 200, orta ve liseye giden için 400, yükseköğrenimdekiler için 600 liradır.
TAILEY	: Toplam aile yardımı=AILEY+COCUKY
MEDNI	: Memurun medeni durumu. Bu değişkenin alacağı değer, eğer memur bekar ise 0 (sıfır), evli ise 1, boşanmış ise 2 olarak belirtilecek ve okutulacaktır.
CALIS	: Eşin çalışıp çalışmadığını belirten değişken. Eş çalışmıyor ise CALIS değişkeninin değeri sıfır (0) dır. Çalışıyor ise 1'dir.
CINS	: Memurun cinsiyeti. CINS=0 ise kadın, CINS=1 ise erkek.
KCOCUK	: Memurun sahip olduğu çocuk sayısı. En fazla 10 çocuk alınmıştır.
EGITIM (L)	: L'nci çocuğun eğitim düzeyi. Bu değişkenin değeri çocuk okula gitmiyor ise 1, ilkokula gidiyorsa 2, orta ya da liseye gidiyorsa 3, yüksek öğretim öğrencisi ise 4 olarak bildirilecektir. L=1,2....., KCOCUK.
GVM	: Gelir vergisi matrahı.
YANOD	: Memurun yan ödemesi.
AILIN	: Aile indirimi.
BORC	: Borçlar.

GELVER : Gelir vergisi.
MALDEN : Dali denge vergisi.
DAMRES : Damga resmi.
YAKYAR : Yakacak yardımı.
GENELT : Genel toplam.
KESINT : Kesinti toplamı.
NETODE : Net ödenecek miktar (aylık).

Akış - Şeması




A
MEDNI
1
-
bekar
15
+
boşanmış
5
evli
0
CALIS
0
=
hayır
AILEY=600.0
5
<>
evet
CINS
0
=
kadın
15
erkek
<>
5
KCOCUK
0
=
yok
15
<>
var
DO 30 L=1,KCOCUK
KEY=EGITIM(L)
KEY
1
2
3
4
COCUKY=COCUKY+100.
COCUKY=COCUKY+200.
COCUKY=COCUKY+400
COCUKY=COCUKY+600
30
15
B


B
TAILEY=AILEY+COCUKY
GVM=MAAST+YANOD-EMK10-AILIN-BORC
GELVER=GVM*0.39
MALDEN=GVM*0.02
DAMRES=(MAAST+YAKYAR+YANOD)*0.004
GENELT=MAAST+EMK18+YAKYAR+TAILEY+YANOD
KESINT=EMK18+EMK10+GELVER+MALDEN+DAMRES+BORC
NETODE=GENELT-KESINT
WRITE
Bordro Çıktı
sı
10
DUR

Program :

```
C  AYLIK BORDRONUN HAZIRLANMASI
       REAL MAAST, KATSAY, MALDEN, KESINT, NETODE
       INTEGER CALIS, CINS, EGITIM (10)
       DIMENSION GOSTER (15,9)
     1 FORMAT (13)
     2 FORMAT (5A4, 2I2, F2.0, 4F7.2, 3I1, I2)
     6 FORMAT (9F7.0)
     7 FORMAT (10I1)
     3 FORMAT (1H1, 55X, 'AYLIK BORDRO', / /,
    ★23X, 'DER', 4X, 'MAAS', 4X, 'AILE', 4X, 'YAN',
    ★4X, 'YAKCAK', 2X, 'BORC', 3X, 'EMEKLI',
    ★4X, 'EMEKLI', 3X, 'GELIR', 3X, 'MALDEN',
    ★2X, 'DAMGA', 5X, 'GENEL', 4X, 'KESINTI',/,
    ★3X, 'ADI SOYADI SICIL NO', 1X, 'KAD', 3X,
    ★'TUTARI', 3X, 'YARDIM', 2X, 'ODEME', 3X,
    ★'YARDIM', 1X, 'TOPLAM', 2X, 'KES.18', 4X,
    ★'KES.10', 2X, 'VERGISI', 2X, 'VERGI.', 2X,
    ★'RESMI', 4X, 'TOPLAMI', 3X, 'TOPLAMI')
     4 FORMAT (/ /, 2X, 5A4, I2, '/', I1, 2X, F8.1,
    ★2X, F5.0, 2X, F6.0, 2X, F5.0, 2X, F6.0, 2X, F7.1,
    ★2X, F7.1, 2X, F7.1, 2X, F6.1, 2X, F6.1,
    ★2X, F8.1, 2X, F8.1,/,
    ★5X, 'NET ODENECEK=', F10.2, 'TL.')
C  BASLIKLARIN YAZDIRILMASI
       WRITE (6,3)
C  MEMUR SAYISI VE GOSTERGE TABLOSUNU OKUTMA
       READ (5,1) N
       READ (5,6) ((GOSTER (I,J), J=1,9), I=1,15)
C  HER MEMUR ICIN VERILERIN OKUTULMASI VE
C  AYLIGININ HESAPLANMASI
       DO 10 M=1,N
       READ (5,2) AD1, AD2, AD3, AD4, AD5, I,J, KATSAY,
      ★YANOD, BORC, YAKYAR, AILIN, MEDNI, CALIS,
      ★CINS, KCOCUK
       IF (KCOCUK.NE.O) READ (5,7) (EGITIM (L),
      ★L=1, KCOCUK)
       MAAST=KATSAY★GOSTER (I,J)
       EMK18=MAAST★0.18
       EMK10=MAAST★0.10
```

```
      AILEY=0.0
      COCUKY=0.0
      IF (MEDNI-1) 15,20,5
   20 IF (CALIS.NE.0) GO TO 25
      AILEY=600.0
      GO TO 5
   25 IF (CINS.EQ.0) GO TO 15
    5 IF (KCOCUK.EQ.0) GO TO 15
      DO 30 L=1, KCOCUK
      KEY=EGITIM (L)
      GO TO (31,32,33,34), KEY
   31 COCUKY=COCUKY+100.0
      GO TO 30
   32 COCUKY=COCUKY+200.0
      GO TO 30
   33 COCUKY=COCUKY+400.0
      GO TO 30
   34 COCUKY=COCUKY+600.0
   30 CONTINUE
   15 TAILEY=AILEY+COCUKY
      GVM=MAAST+YANOD-EMK10-AILIN-BORC
      GELVER=GVM*0.39
      MALDEN=GVM*0.02
      DAMRES=(MAAST+YAKYAR+YANOD)*0.004
      GENELT=MAAST+EMK18+YAKYAR+TILEY+
     *YANOD
      KESINTI=EMK18+EMK10+GELVER+MALDEN+
     *DAMRES+BORC
      NETODE=GENELT-KESINT
      WRITE (6,4) AD1, AD2, AD3, AD4, AD5, I,J, MAAST,
     *TAILEY, YANOD, YAKYAR, BORC, EMK18, EMK10,
     *GELVER, MALDEN, DAMRES, GENELT, KESINT,
     *NETODE
   10 CONTINUE
      STOP
      END
```

Yazılan programda da görüldüğü gibi bordro için gerekli başlıklar yazıldıktan sonra program girdileri olarak önce memur sayısı N ve göstergeleri belirten tablo okutulmaktadır. Daha sonra ise her personel (memur) ile ilgili olan veriler okutulmaktadır. Personelin adı, soyadı ve silcil numarası birlikte alfa-nümerik değişken olarak 20 sü-

tunda okutulmaktadır. Personelin derece ve kademesi (I ve J)'de ayrıca okutulmakta ve her personel için buna göre gösterge tablosundan memurun göstergesi belirlenmektedir. Diğer yandan bordro için gerekli olan bilgiler : yan ödeme, yakacak yardımı, borçlar, aile indirimi gibi değerler her memur için veri olarak okutulmaktadır. Aynı şekilde personelin özel durumunu belirten diğer bilgilerde girdi olarak verilmektedir. Bunlar daha öncede belirtildiği gibi kodlanarak belirtilmektedir. Örneğin MEDNI değişkeninin değeri, memur bekar ise 0, evli ise 1 ve boşanmış ise 2 olarak veri kartına delinecektir.

Programın gerektirdiği verileri kartlara aktarırken 2 FORMAT deyiminde belirttiği gibi bir memur ile ilgili tüm veriler çocukların eğitim düzeyine ilişkin olanlar hariç bir tek kartta belirtilmektedir. Böyle bir kartta;

1 — — 20.	sütunlarda	: Memurun adı, soyadı ve sicil numarası,
21 — — 22.	»	: derecesi,
23 — — 24.	»	: kademesi,
25 — — 26.	»	: katsayı,
27 — — 33.	»	: yanödemesi,
34 — — 40.	»	: borcu,
41 — — 47.	»	: yakacak yardımı,
48 — — 54.	»	: aile indirimi bildirilecektir.
55.	sütunda	: Medeni durumu,
56.	»	: Eşinin çalışıp çalışmadığı,
57.	»	: cinsiyeti,
58 — — 59.	sütunlarda	: çocuk sayısı

Çocuğu olan personel için çocukların eğitim durumu her çocuk için ayrı bir sütunda 7 FORMAT'a göre bir karta delinecektir.

Örnek 5 : Anket Sonuçlarının Değerlendirilmesi

Bilgi toplama aracı olarak sık sık başvurulan anketlerin sonuçlarını değerlendirmek için hazırlanacak program, anketin biçimi ve soruların türüne göre değişecektir. Bu konuya bir örnek olarak renkli televizyon yayınlarının gerekliliği konusunda bilgi edinmek için hazırlanan aşağıdaki anket örneğini alalım. Hemen belirtilmelidir ki aşağıdaki küçük anket yalnızca program hazırlama amacı ile hazırlanmıştır.

Renkli Televizyon Yayınları ile İlgili Ankent Formu :

(Lütfen aşağıdaki soruları uygun yanıtın karşısına X işareti koyarak yanıtlayınız.)

1. Cinsiyeti :
   a) Erkek b) Kadın

2. Yaşı :
   a) 18'den aşağı. b) 19 - - 39 arası c) 40 - - 59 arası
   d) 60 ve yukarısı.

3. Mesleği :
   a) Çiftçi b) Memur c) İşçi d) Ev Kadını e) Emekli
   f) Öğrenci g) Serbest Meslek h) Başka (belirtiniz).....

4. Aylık geliri :
   a) 15000 TL'den az b) 16000 - 30000 TL c) 31000 - 45000 TL
   d) 46000 - 60000 TL e) 61000 - 75000 TL f) 75000 TL'den fazla

5. Siyah-beyaz televizyonunuz var mı?
   a) Evet b) Hayır

6. Hiç renkli televizyon yayını izlediniz mi?
   a) Evet b) Hayır

7. Sizce ülkemizde renkli televizyon yayını gerekli midir?
   a) Evet b) Hayır c) Bilmiyorum

Not : Eğer yanıtınız "evet" ise aşağıdaki soruyu yanıtlayınız. Aksi durumda atlayınız.

8. Sizce renkli televizyon yayın süresi nasıl olmalıdır?
   a) Tüm yayın renkli olmalıdır.
   b) Her gün bir kaç saat renkli diğeri siyah-beyaz olmalıdır.
   c) Her hafta bir kaç saat renkli diğeri siyah-beyaz olmalıdır.
   d) Kesin bir fikrim yok.

Yukarıda programlama örneği amacı ile hazırlanan anket formuna verilecek yanıtları değerlendirmek için aşağıdaki kodlama biçimini kullanalım.

Anket sonuçlarının kodlanması :

Sorular :

1. Cinsiyet durumu
   a) 1 b) 2 0 Eğer işaretsiz ise.

2. Yaş durumu

a) 1 b) 2 c) 3 d) 4 0 Eğer işaretsiz ise.

3. Mesleği

a) 1 b) 2 c) 3 d) 4 e) 5 f) 6
g) 7 h) 8 0 Eğer işaretsiz ise.

4. Aylık geliri

a) 1 b) 2 c) 3 d) 4 e) 5
f) 6 0 Eğer işaretlenmemiş ise.

5. Siyah - beyaz TV

a) 1 b) 2 o İşaretsiz ise.

6. Renkli TV yayını izleme

a) 1 b) 2 o İşaretsiz ise.

7. Renkli TV gerekliliği

a) 1 b) 2 c) 3 0 Eğer işaretlenmemiş ise.

8. Renkli TV yayın süresi

a) 1 b) 2 c) 3 d) 4 0 Eğer işaretlenmemiş ise.

Hazırladığımız anketi rastgele seçilmiş bir gruba uyguladığımızı varsayalım. Anket sonuçlarının değerlendirilmesi sonucu aradığımız bilgileri sorular biçiminde aşağıdaki gibi belirtelim.

Anket Sonucu İstenilen Bilgiler :

	İlişkili Anket Sorusu	Madde Kodu
1. Kaç erkek renkli TV yayınına taraftardır?	1	1
	7	1
2. Kaç kadın renkli TV yayınını gerekli görmemektedir?	1	2
	7	2
3. Aylık geliri 15000 TL'den aşağı olan siyah - beyaz TV sahibi kaç kişi renkli TV yayınını gerekli görmektedir?	4	1
	5	1
	7	1
4. Renkli TV yayını izlemiş 18 yaşından küçük kaç erkek öğrenci ülkemizde renkli TV yayınını gerekli görmektedir?	6	1
	2	1
	1	1
	7	1

5.	Aylık geliri 75000 TL'den fazla olan serbest meslek sahibi kadınların kaç tanesi hergün bir kaç saat renkli TV yayını istemektedir?	4 3 1 7 8	6 7 2 1 2

Kuşkusuz yukarıda belirtildiği gibi bir kaç soruya dayalı olarak bir bilgi sahibi olabileceğimiz gibi her sorunun her maddesi de bize ayrıca çeşitli bilgiler sağlayacaktır. Örneğin, 2. sorunun her maddesine verilen yanıta göre 150 kişilik grubun yaş dağılımını, 4. soruya göre gelir dağılımını bulabileceğimiz gibi kaç kişinin televizyonu yok, kaç kişi hiç renkli televizyon yayını izlememiştir v.b. konularla da ilgili olarak bilgi edinebiliriz. Bu bilgiler için her sorunun her maddesine verilen yanıtları toplamak yeterli olacaktır.

Şimdiye kadar yapılan açıklamalara dayalı olarak verilen örnek anketin 150 kişilik gruba uygulanması sonucu elde edilen verilerin dökümünü yapacak ve aradığımız sorulara yanıt verecek bilgileri elde etmemizi sağlayacak bir program hazırlanabilir.

Değişkenler :

I : Anketteki soru numarası. Örneğimizde en son numara 8'dir.

J : Ankete yanıt veren birey sayısı. Örneğimizde bu 150'dir.

K : Anketteki soru maddeleri ya da şıkları. Örneğimizde K'nin değeri 2 ile 8 arasında değişmektedir. Diğer yandan yanıtlanmamış soruların boş bırakıldığını belirtmek için kullandığımız 0 rakkamınıda göz önüne alırsak K'nin en son değeri $8+1=9$ olacaktır.

SORU (I,J) : J'nci anketteki I'nci soru. $I=1,2,\ldots,8$ ve $J=1,2,\ldots,150$.

TOPLA (K) : K'nci madde ya da şıkların toplamı. Yani K'nci maddeye verilen yanıtlar toplamı.

BILGI1 : Anket sonucu istediğimiz bilgiyi sağlayan 1. soruya verilen yanıt toplamı. Yani Kaç erkek renkli TV yayınına taraftardır?

BILGI2 : Bilgi sağlayan 2. soruya verilen yanıt toplamı.

:

:

BILGI5 : Anket sonucu istediğimiz en son bilgiyi sağlayan 5. soruya verilen yanıt toplamı.

M : BILGI değişkenlerini 1,2, ... 5 biçiminde sıralamak için kullanılan değişken.

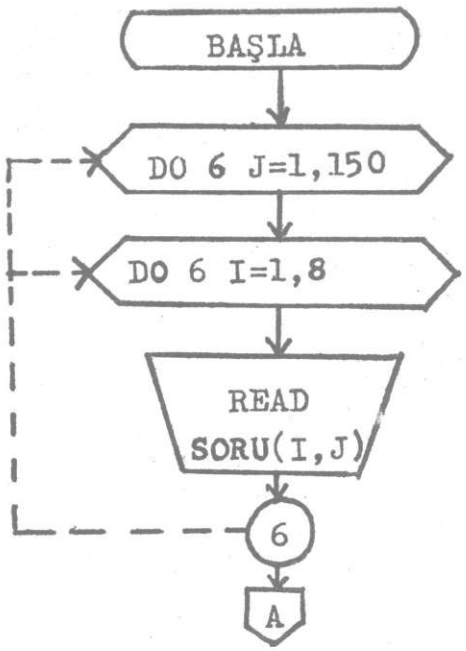
Programda kullanılacak değişkenler böylece belirlendikten sonra izleyeceğimiz yöntemi kısaca açıklayalım :

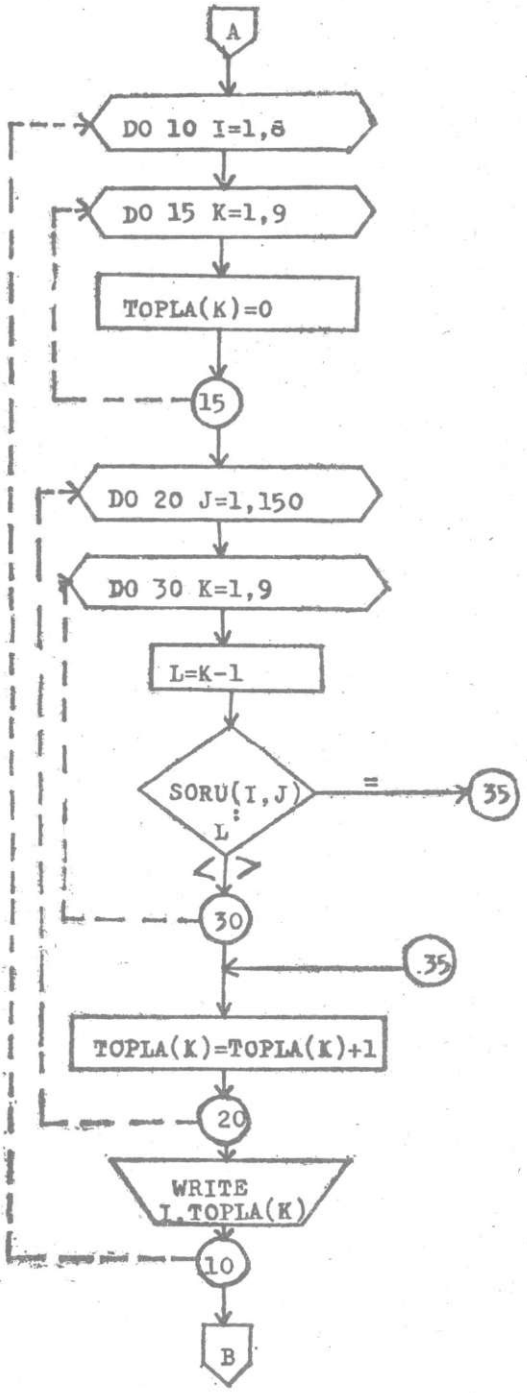
1. Her anket formuna verilen yanıtlar belirtilen kodlama biçimine ve 1 FORMAT'a göre kartlara delindikten sonra bilgisayara okutulacaktır, READ (5,1).

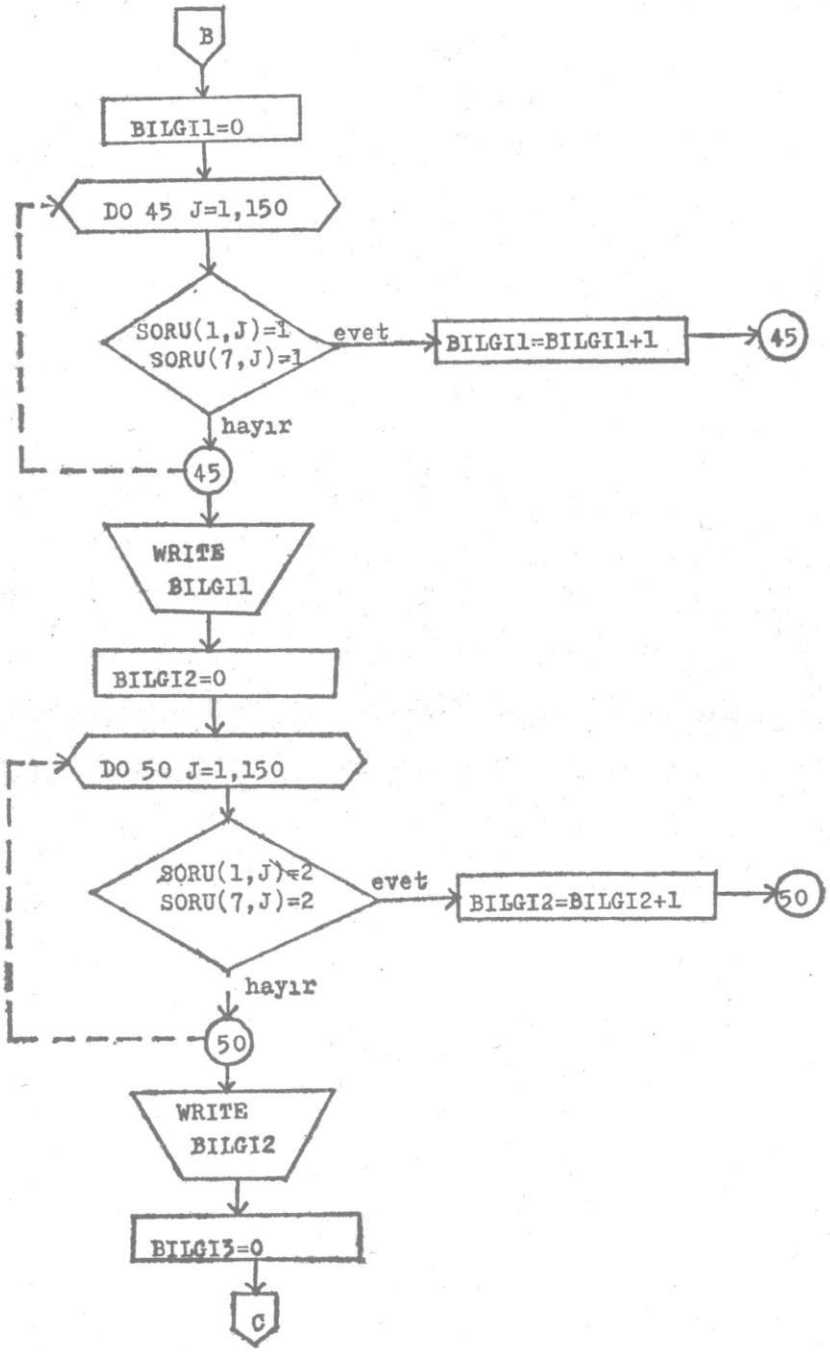
2. Okutma işleminden sonra her sorunun her maddesine verilen yanıtlar toplanacaktır.

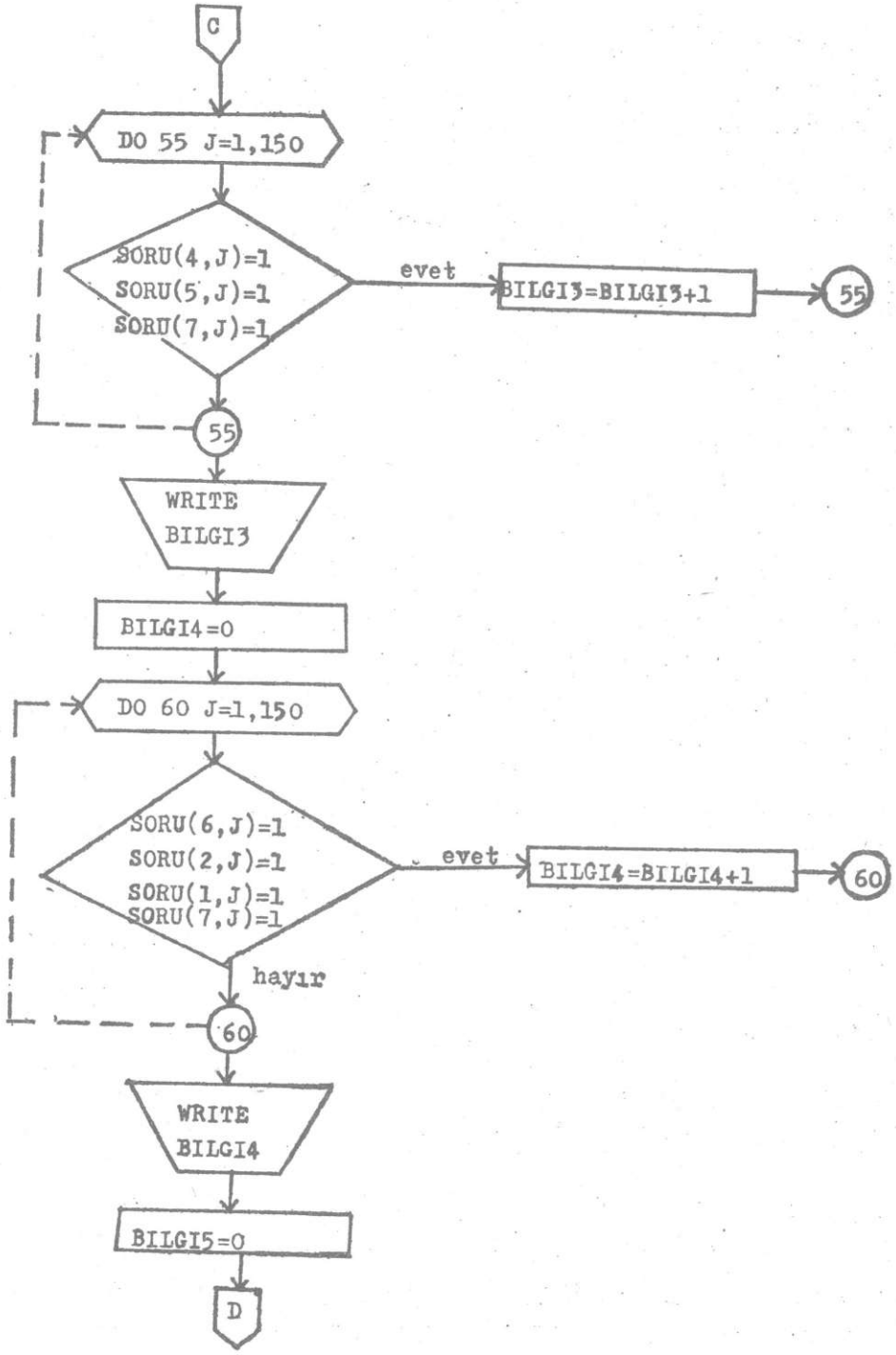
3. Daha sonra çeşitli sorulara dayalı olarak istenilen 5 ayrı soruya yanıt verecek bilgiler BILGI1, BILGI2,...,BILGI5 için ilgili anket sorularının ilgili maddeleri ayrı ayrı toplanacak ve sonuçlar yazdırılacaktır.

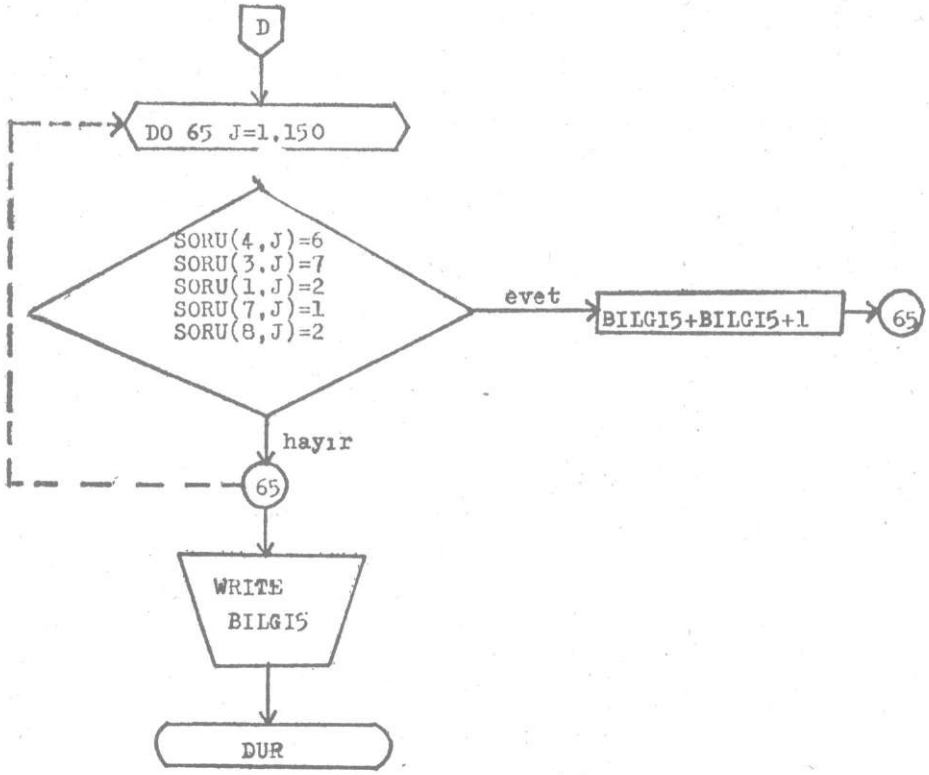
Buna göre akış-şeması şöyle olacaktır.




A
DO 10 I=1,8
DO 15 K=1,9
TOPLA(K)=0
15
DO 20 J=1,150
DO 30 K=1,9
L=K-1
SORU(I,J)
L
=
35
<>
30
35
TOPLA(K)=TOPLA(K)+1
20
WRITE
J,TOPLA(K)
10
B


B
BILGI1=0
DO 45 J=1,150
SORU(1,J)=1
SORU(7,J)=1
evet
BILGI1=BILGI1+1
45
hayır
45
WRITE
BILGI1
BILGI2=0
DO 50 J=1,150
SORU(1,J)=2
SORU(7,J)=2
evet
BILGI2=BILGI2+1
50
hayır
50
WRITE
BILGI2
BILGI3=0
C


C
DO 55 J=1,150
SORU(4,J)=1
SORU(5,J)=1
SORU(7,J)=1
evet
BILGI3=BILGI3+1
55
55
WRITE
BILGI3
BILGI4=0
DO 60 J=1,150
SORU(6,J)=1
SORU(2,J)=1
SORU(1,J)=1
SORU(7,J)=1
evet
BILGI4=BILGI4+1
60
hayır
60
WRITE
BILGI4
BILGI5=0
D



Yukarıdaki akış şemasına dayalı olarak program :

```
C  ANKET SONUCLARININ DEGERLENDIRILMESI
C  DEGISKEN TIPLERI VE BOYUT BILDIRME
       INTEGER SORU (8,150), TOPLA (9), BILGI1, BILGI2,
     ★BILGI3, BILGI4, BILGI5
     1 FORMAT (8I1)
     2 FORMAT (1H1, 30X, 'ANKET SONUCLARI', / / /,
     ★5X, 'SORU MADDELERININ',/, 8X, KODU : bbbbb0
     ★bbbbb1bbbbb2bbbbb3bbbbb4bbbbb5bbbbb6bbbbb7
     ★bbbbb8', / /, 5X, 'SORU NO', / /)
     3 FORMAT (10X, I2, 9I6,/)
     4 FORMAT (15X, 'ISTENILEN BILGILER', / / /,
     ★10X, 'BILGI SORU NO.', 15X, 'YANIT SAYISI', / /)
     5 FORMAT (15X, 'BILGI', I1, 15X, I7,/ /)
       DO 6 J=1,150
     6 READ (5,1) (SORU (I,J), I=1,8)
```

```
C  BASLIK YAZDIRMA
       WRITE (6,2)
C  HER MADDENIN DOKUMUNU YAPMAK
       DO 10 I=1,8
       DO 15 K=1,9
   15  TOPLA (K) =0
       DO 20 J=1,150
       DO 30 K=1,9
       L=K-1
       IF (SORU (I,J).EQ.L) GO TO 35
   30  CONTINUE
   35  TOPLA (K) =TOPLA (K) +1
   20  CONTINUE
       WRITE (6,3) I, (TOPLA (K), K=1,9)
   10  CONTINUE
       WRITE (6,4)
C  ISTENILEN BILGILERI TOPLAMAK VE YAZMAK
       M=1
       BILGI1=0
       DO 45 J=1,150
       IF(SORU(1,J).EQ.1.AND.SORU(7,J).EQ.1) BILGI1=
      ★BILGI1+1
   45  CONTINUE
       WRITE (6,5) M,BILGI1
       M=M+1
       BILGI2=0
       DO 50 J=1,150
       IF(SORU(1,J).EQ.2.AND.SORU(7,J).EQ.2) BILGI2=
      ★BILGI2+1
   50  CONTINUE
       WRITE (6,5) M, BILGI2
       M=M+1
       BILGI3=0
       DO 55 J=1,150
       IF (SORU (4,J).EQ.1.AND.SORU (5,J).EQ.1.AND.SORU
      ★(7,J).EQ.1) BILGI3=BILGI3+1
   55  CONTINUE
       WRITE (6,5) M, BILGI3
       M=M+1
       BILGI4=0
       DO 60 J=1,150
```

```
      IF (SORU (6,J).EQ.1.AND.SORU (2,J).EQ.1.AND.SORU
     ★(1,J).EQ.1.AND.SORU (7,J).EQ.1) BILGI4=BILGI4+1
60    CONTINUE
      WRITE (6,5) M, BILGI4
      M=M+1
      BILGI5=0
      DO 65 J=1,150
      IF (SORU (4,J).EQ.6.AND.SORU (3,J).EQ.7.AND.SORU
     ★(1,J).EQ.2.AND.SORU (7,J).EQ.1.AND.SORU (8,J).EQ.
     ★2) BILGI5=BILGI5+1
65    CONTINUE
      WRITE (6,5) M, BILGI5
      STOP
      END
```

Yukarıda verilen programda 1 FORMAT'ında belirttiği gibi, 8 soru içeren her anket formunun yanıtları aynı karta kodlanmış olarak delinecektir. Böylece 150 kişilik grup için 150 veri kartı kullanılacaktır. Yine aynı FORMAT'ın belirttiği gibi her anket formunun yanıtları kodlanmış olarak 1 sütunluk yerlere peşpeşe kartlara delinecektir. Örneğin, bir anket formunda;

1. soru için a, kodu : 1
2. soru için c, kodu : 3
3. soru için f, kodu : 6
4. soru için a, kodu : 1
5. soru için b, kodu : 2
6. soru için a, kodu : 1
7. soru için b, işaretlenmiş ise (kodu : 2) ve
8. soru işaretlenmemiş yani boş bırakılmış olduğunu varsayalım. (kodu : 0).

Başlangıçta verilen kodlama biçimi anımsanırsa bu anket formunun sonuçları bir tek karta aşağıdaki gibi peş peşe ilk 8 sütuna yazılacaktır :

13612120

Örnek 6 : Doğrusal Progrmlama Problemlerinin Simplex Yöntemi ile Çözümü

Doğrusal programlama problemlerinin çözümü için Simplex yöntemini kullanan bir program hazırlamak isteyelim. Programın açıkla-

masını ve anlaşılmasını kolaylaştırmak için Simplex çözüm yöntemini aşağıdaki örnekle açıklayalım.

$$\begin{aligned} 10X_1 + 2X_2 + X_3 &\leqslant 100 \\ 3X_1 + 13X_2 + 4X_3 &\leqslant 150 \\ 2X_1 + 3X_2 + 12X_3 &\leqslant 120 \end{aligned}$$

Bu sınırlılıklar altında ve değişkenlerin negatif olmaması koşulu ile, yani; $X_i \geqslant 0 \qquad (i=1,2,3,....)$

$$Z_{max} = 5X_1 + 7X_2 + 6X_3$$

amaç fonksiyonunu en büyükleyelim (maksimize edelim).

Simplex çözüm yöntemi aşağıdaki aşamaları izleyecektir :

1. Boş değişkenler ekleyerek eşitsizlikleri eşitlik haline çevirmek ve ilk temel mümkün çözümü sağlamak için gerekirse yapay değişkenler eklemek.

$$\begin{aligned} 10X_1 + 2X_2 + X_3 + X_4 &= 100 \\ 3X_1 + 13X_2 + 4X_3 + X_5 &= 150 \\ 2X_1 + 3X_2 + 12X_3 + X_6 &= 120 \end{aligned}$$

Yukarıdaki eşanlı eşitliklerdeki X_4, X_5 ve X_6 kullanılmayan kapasiteyi temsil eden boş değişkenler olduğundan amaç fonksiyonundaki katsayıları sıfır (0) dır. Yani;

$$Z_{max} = 5X_1 + 7X_2 + 6X_3 + 0X_4 + 0X_5 + 0X_6$$

Bu problemde boş değişkenlerin eklenmesi ile ilk temel mümkün çözüm bulunabilmekte ve yapay değişken eklemeğe gerek kalmamaktadır. Şöyleki değişken sayısı (N) ve eşitlik sayısı (M) farkı, $N-M=6-3=3$ bulunur. $X_1=X_2=X_3=0$ olarak alınırsa ilk temel çözüm için simplex tablosu aşağıdaki gibi görünecektir.

Tablo 1

C_j			5	7	6	0	0	0	
C_i	Çözüm	B	X_1	X_2	X_3	X_4	X_5	X_6	T
0	X_4	100	10	2	1	1	0	0	50
0	X_5	150	3	[13]	4	0	1	0	11.538
0	X_6	120	2	3	12	0	0	1	40
	Z_j	0	0	0	0	0	0	0	
	C_j-Z_j		5	7	6	0	0	0	

Yukarıdaki tabloda amaç fonksiyonundaki kat sayılar tablonun en üst sırasında C_j olarak gösterilmektedir. B değişkeni sütunundaki değerler eşanlı sınırlılık eşitliklerindeki sağ tarafta yer alan değerlerdir. Tablodaki diğer sayılar ise sınırlılık eşitliklerindeki X'lerin katsayılarını göstermektedir. Tablodaki bu katsayıları A_{ij} simgesi ile gösterelim. Örneğin, $A_{12}=10$ ve $A_{23}=4$ gibi.

2. Yukarıdaki tablonun Z_j sırası

$$Z_j=\sum_{i=1}^{m} A_{ij}\, C_i \qquad \text{(tüm j'ler için)}$$

biçiminde hesaplanır. Örneğin;

B sütunu için $Z_j=100(0)+150(0)+120(0)=0$

X_1 sütunu için $Z_j=10(0)+3(0)+2(0)=0$ gibi

3. Amaç fonksiyonuna katkısı en fazla olan değişkeni belirlemek. Yani, C_j-Z_j farkı en fazla olan sütunu saptamak. C_j-Z_j farkları, tablo 1'de de görüldüğü gibi tablonun en üst sırasında yer alan amaç fonksiyonunun katsayıları C_j ile hesaplanan Z_j farkına eşittir. Örneğin;

X_1 sütunu için $C_j-Z_j=5-0=5$
X_2 sütunu için $C_j-Z_j=7-0=7$ gibi.

Tablodaki C_j-Z_j değerlerine bakıldığında katkısı en fazla olan değişken X_2'dir. Dolayısiyle çözüme alınacak olan değişkende X_2 olmalıdır.

4. Çözümden çıkarılacak, yani katkısı en az olan değişkeni belirlemek. Bunun için de tablonun en sağında T sütunu değerleri kullanılmaktadır. T değeri en az olan çözümdeki değişken çıkarılacak olan değişkendir. T değerleri, B sütunundaki değerlerin çözüme alınacak olan değişkenin yer aldığı sütundaki katsayılara bölünmesi ile elde edilmektedir. Örneğin tablo 1'de;

Birinci sırada $T = 100/\ 2 = 50$
İkinci sırada $T = 150/13 = 11.538$
Üçüncü sırada $T = 120/\ 3 = 40$ bulunur.

Böylece çözümden çıkarılacak değişken boş değişken olan X_5'dir. Çünkü bu sırada T'nin değeri en küçük (11.538) olmaktadır.

5. Çözüme girecek değişkeni tabloya almak ve çıkacak olanı çıkarmak için gerekli sıra işlemlerini yapmak. Bunun için :

a) Çözüme girecek yeni değişken (X_2) sırası, çıkacak olan değişken (X_5) sırasının yerini alacaktır. Yer değiştirme işlemi için çözüme giren değişken sütunu ile çözümden çıkacak değişken sırasının kesiştiği elemanın (pivot eleman) katsayısı 1 ve çözüme giren değişken sütununun diğer elemanlarının katsayısı sıra işlemleri ile sıfır (0) yapılır. Tablo 1'de pivot eleman katsayısı 13 olan ve dörtgen içine alınan elemandır. Bu katsayının 1 olması için sıranın tüm elemanlarının 13 ile bölünmesi yeterli olacaktır. Böylece yeni sıra;

7 X_2 11.538 0.231 1 0.308 0 0.077 0

olarak bulunur. Bu sıradaki birinci eleman 7, çözüme giren X_2'nin amaç fonksiyonundaki katsayısıdır.

b) Çözüme giren değişken sütunundaki diğer katsayıların sıfır (0) yapılabilmesi ise, yeni elde edilen sıranın elemanlarının o sütundaki katsayılarla çarpılması ve sonuçların sıra elemanlarından çıkartılması ile yapılır.

Örneğin, X_4 değişkeninin eski sırası aşağıdaki gibi değişecektir :

100	10	2	1	1	0	0
— 2(11.538)	— 2(0.231)	— 2(1)	— 2(0.308)	— 2(0)	— 2(0.077)	— 2(0)
76.924	9.538	0	0.384	1	—0.154	0

Aynı şekilde X_6 değişkeninin yeni sırası;

120	2	3	12	0	0	1
— 3(11.538)	— 2(0.231)	— 3(1)	— 3(0.308)	— 3(0)	— 3(0.077)	— 3(0)
85.386	1.307	0	11.076	0	—0.231	1

Böylece X_2 ve X_5 değişkenlerinin çözümde yer değişimi sonucu elde edilen ikinci tablo aşağıda verilmiştir. Z_j ve $C_j - Z_j$ değerleride önceden belirtildiği gibi hesaplanmıştır. Örneğin, B sütunu için $Z_j = o(76.924) + 7(11.538) + 0(85.386) = 80.766$

Tablo 2

C_j :			5	7	6	0	0	0	
C_i	Çözüm	B	X_1	X_2	X_3	X_4	X_5	X_6	T
0	X_4	76.924	9.538	0	0.384	1	—0.154	0	200.3
7	X_2	11.538	0.231	1	0.308	0	0.077	0	37.5
0	X_6	85.386	1.307	0	11.076	0	—0.231	1	7.709
	Z_j	80.766	1.617	7	2.156	0	0.539	0	
	$C_j - Z_j$		3.383	0	3.844	0	—0.539	0	

6. İkinci tabloda da daha önceden olduğu gibi çözüme girecek ve çıkacak olan değişkenler belirlenir ve yer değişimi için gerekli sıra işlemleri yapılır. Elde edilen yeni değerlere göre tekrar bir tablo kurulur. Şimdiye kadar açıklanan işlem aşamaları, $C_j - Z_j$ farkı tüm değişkenler için sıfır (0) ya da negatif (—) oluncaya kadar devam eder. $C_j - Z_j$ farklarının tüm değişkenler için 0 ya da negatif olduğu aşama amaç fonksiyonunu en büyükleyen (maksimize eden) optimal çözümdür. En küçükleyen (minimizasyon) problemlerinde ise amaç fonksiyonunun negatifinin en büyüklemesi (maksimizasyonu) ile aynı yöntemle çözüme ulaşılır.

Örnek alınan problemi çözmek için işleme devam edersek, tablo 2'ye göre çözüme girecek değişken X_3 ve çıkacak olan ise X_4'dır. Sıra değiştirme işleminden sonra tablo 3 aşağıdaki gibi olacak.

Tablo 3

	C_j :		5	7	6	0	0	0	
C_i	Çözüm	B	X_1	X_2	X_3	X_4	X_5	X_6	T
0	X_4	73.964	[9.493]	0	0	1	—0.146	—0.035	7.791
7	X_2	9.164	0.195	1	0	0	0.083	—0.028	47.0
6	X_3	7.709	0.118	0	1	0	—0.021	0.090	65.3
	Z_j	110.402	2.073	7	6	0	0.455	0.344	
	$C_j - Z_j$		2.927	0	0	0	—0.455	—0.344	

Tablo 3'de çözüme girecek olan değişken X_1 çıkacak olan ise X_4'dür. Yer değiştirme işleminden sonra tablo 4 aşağıdaki gibi olacaktır.

Tablo 4

	C_j :		5	7	6	0	0	0	
C_i	Çözüm	B	X_1	X_2	X_3	X_4	X_5	X_6	T
5	X_1	7.791	1	0	0	0.105	—0.015	—0.004	
7	X_2	7.645	0	1	0	—0.020	0.086	—0.027	
6	X_3	6.790	0	0	1	—0.012	—0.019	0.090	
	Z_j	133.21	5	7	6	0.313	0.413	0.331	
	$C_j - Z_j$		0	0	0	—0.313	—0.413	—0.331	

Tabloda da görüldüğü gibi tüm $C_j - Z_j$ farkları sıfır (0) ya da negatiftir. Optimal çözüm ise :

$X_1 = 7.791$

$X_2 = 7.645$

$X_3 = 6.790$

Şimdiye kadar verilen açıklamalara dayalı olarak simplex yöntemi için hazırlanacak programın akış şeması çizilebilir. Ancak daha önce programda kullanılacak değişkenleri belirtelim.

Değişkenler :

N	: Problemdeki tüm değişkenler sayısı (boş ve yapay değişkenler dahil).
M	: Problemdeki eşitlik sayısı.
KX	: Problemdeki gerçek değişkenler sayısı (boş ve yapay değişkenler hariç).
C(J)	: Amaç fonksiyonundaki değişken katsayıları. Yani C_j
A(I,J)	: Eşitliklerdeki değişken katsayıları, yani A_{ij} I=1,2,...,M ve J=1,2.....,N
B(I)	: Eşanlı sınırlılık eşitliklerindeki B_i değerleri.
KCOZUM (I)	: Çözüme giren değişkenlerin sırasını belirten bir değişken dizisi. Örneğin, KCOZUM (1)=4 ise çözüm tablosunun 1. sırasındaki değişken problemdeki 4. değişkendir, yani X_4'tür.
CI(I)	: Çözüme giren değişkenlerin amaç fonksiyonundaki katsayıları. Yani tablolardaki C_i sütunu.
Z(J)	: Değişkenlerin tablodaki Z_j değerleri.
CZF(J)	: $C_j - Z_j$ farkları.
ITE	: Tabloları tekrarlama (iterasyon) sayısı.
BUYUK	: En büyük $C_j - Z_j$ farkını belirtmektedir.
IGELEN	: Çözüme girecek değişkenin sütununu göstermektedir.

TED : Tablonun T (teta) sütunundaki değerleri belirtmektedir.

EKTED : TED değerlerinin en küçüğünü belirtmektedir.

IGIDEN : Çözümden çıkarılacak değişkenin sırasını belirtir.

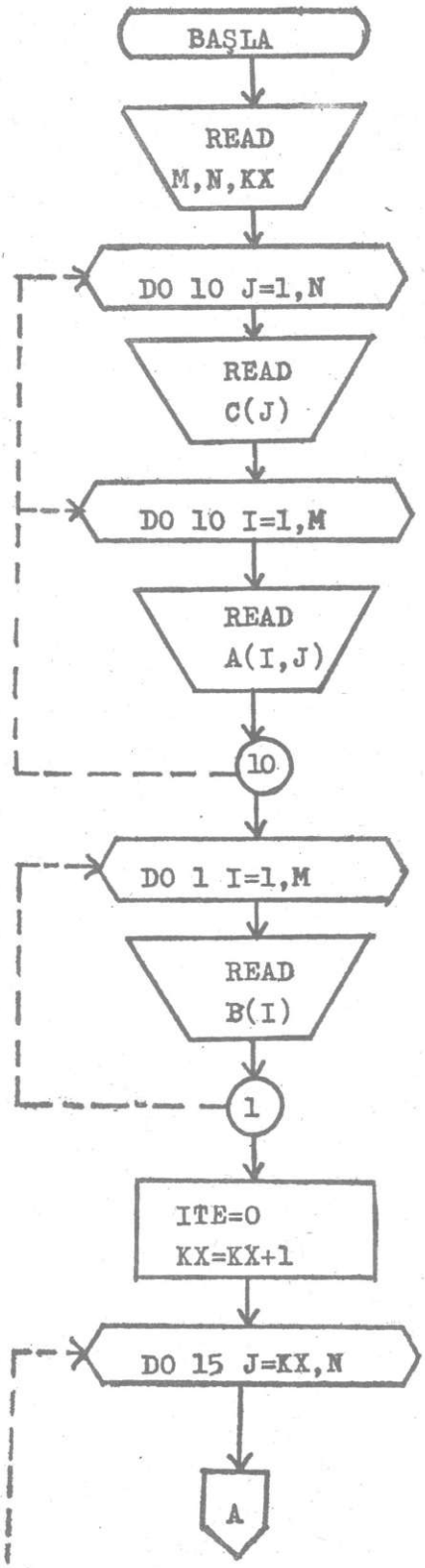
PIVOT : Pivot elemanın katsayısını göstermektedir.

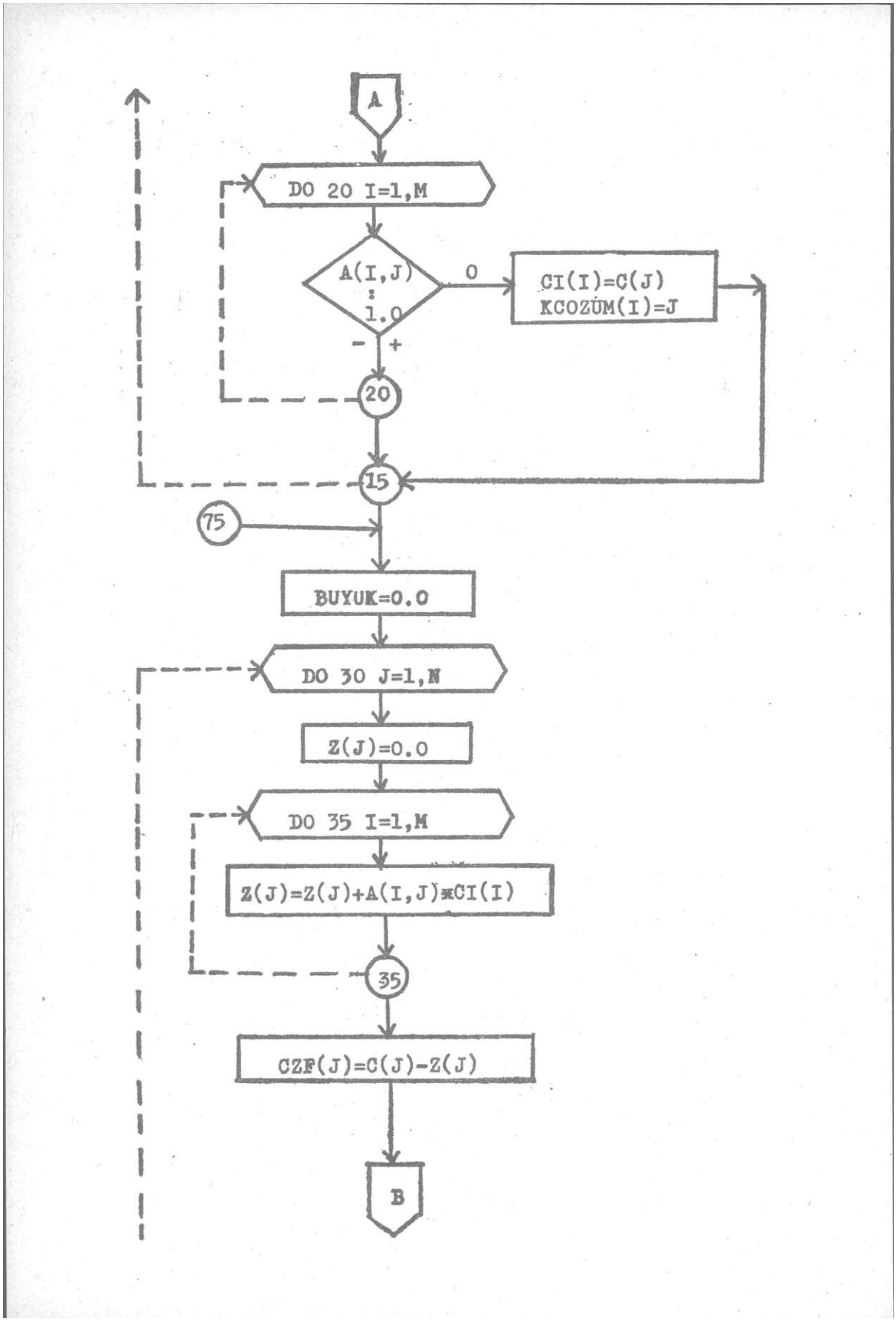
SUTUN : Çözüme giren değişkenin bulunduğu sütundaki katsayısını göstermektedir.

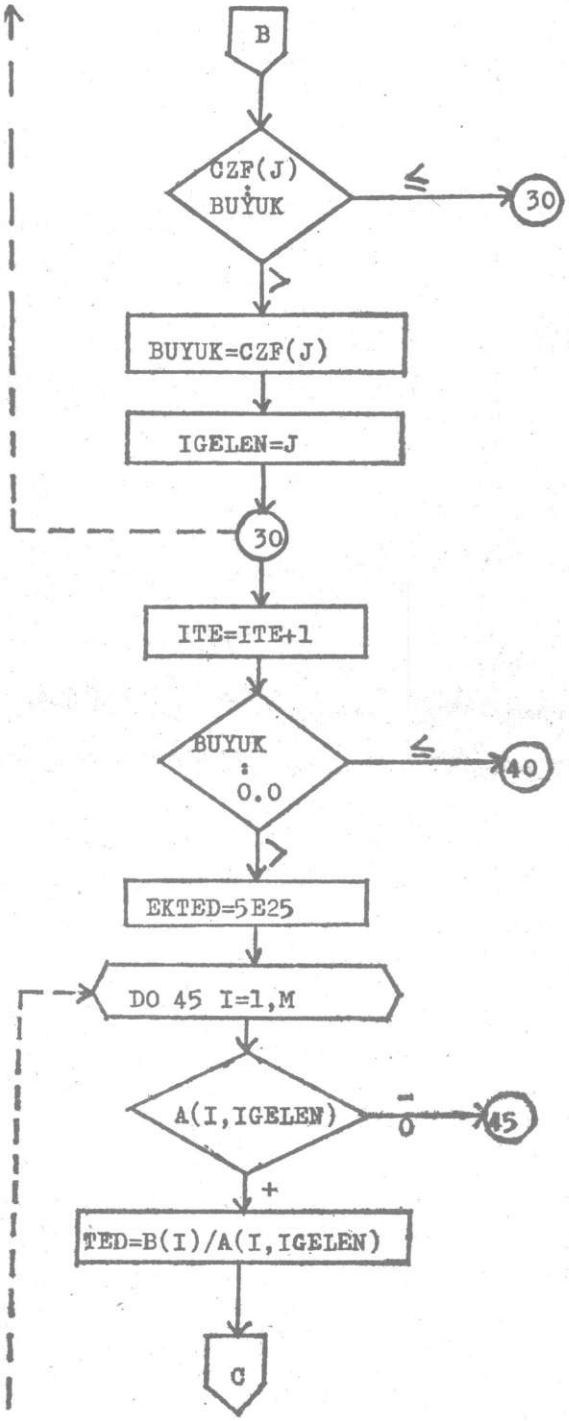
SIRA : Tabloların yeni sıralarını hazırlamak için kullanılan bir değişken.

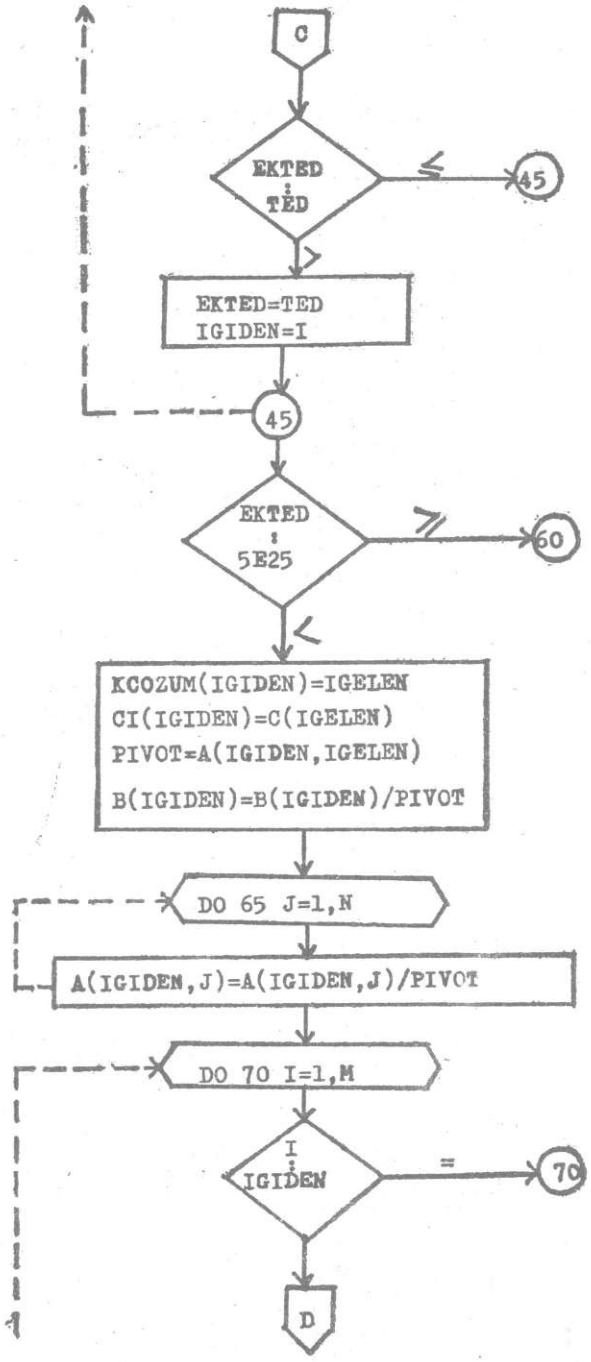
ZMAX : En büyüklenecek (maksimize edilecek) amaç fonksiyonunun çözü sonucu bulunan değeri.

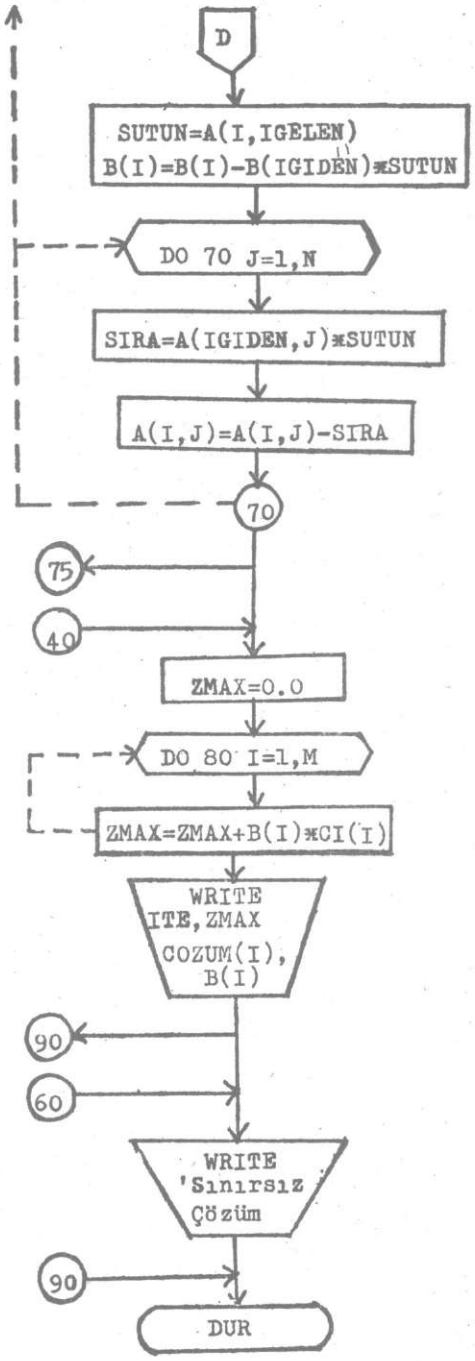
Akış şeması ve program bitişik sayfalarda verilmiştir.


BAŞLA
READ
M,N,KX
DO 10 J=1,N
READ
C(J)
DO 10 I=1,M
READ
A(I,J)
10
DO 1 I=1,M
READ
B(I)
1
ITE=0
KX=KX+1
DO 15 J=KX,N
A


A
DO 20 I=1,M
A(I,J)
:
1.0
0
CI(I)=C(J)
KCOZÜM(I)=J
-
+
20
15
75
BUYUK=0.0
DO 30 J=1,N
Z(J)=0.0
DO 35 I=1,M
Z(J)=Z(J)+A(I,J)*CI(I)
35
CZF(J)=C(J)-Z(J)
B


B
CZF(J)
:
BUYUK
≤
30
>
BUYUK=CZF(J)
IGELEN=J
30
ITE=ITE+1
BUYUK
:
0.0
≤
40
>
EKTED=5E25
DO 45 I=1,M
A(I,IGELEN)
=
0
45
+
TED=B(I)/A(I,IGELEN)
C


C
EKTED
:
TED
<
45
>
EKTED=TED
IGIDEN=I
45
EKTED
:
5E25
>
60
<
KCOZUM(IGIDEN)=IGELEN
CI(IGIDEN)=C(IGELEN)
PIVOT=A(IGIDEN,IGELEN)
B(IGIDEN)=B(IGIDEN)/PIVOT
DO 65 J=1,N
A(IGIDEN,J)=A(IGIDEN,J)/PIVOT
DO 70 I=1,M
I
:
IGIDEN
=
70
D


D
SUTUN=A(I,IGELEN)
B(I)=B(I)-B(IGIDEN)*SUTUN
DO 70 J=1,N
SIRA=A(IGIDEN,J)*SUTUN
A(I,J)=A(I,J)-SIRA
70
75
40
ZMAX=0.0
DO 80 I=1,M
ZMAX=ZMAX+B(I)*CI(I)
WRITE
ITE,ZMAX
COZUM(I),
B(I)
90
60
WRITE
'Sınırsız
Çözüm
90
DUR

Program : Boyutlar (10×30) olarak alınmıştır.

```
C  DOGRUSAL PROGRAMLAMA
C  SIMPLEX YONTEMI
       DIMENSION C(30), A(10,30), B(10), CI(10),
       DIMENSION Z(30), CZF (30), KCOZUM (10)
C  ESITLIK VE DEGISKEN SAYILARINI OKUTMA
       READ (5,1) M,N,KX
    1  FORMAT (3I5)
    2  FORMAT (11F8.2)
   55  FORMAT (10F8.2)
    3  FORMAT (1H1, 15X, 'COZUM', / / /,
      15X, 'TEKRARLAMA SAYISI=', I5, / /,
      25X, 'AMAC FONKSIYONU DEGERI=', F12.2, / /,
      35X, 'DEGISKENLER', 3X, 'DEGERLERI')
    4  FORMAT (5X, I5, 10X, F12.2)
    5  FORMAT(15X,'AMAC FONKSIYONU SINIRLI DEGIL')
C  KATSAYILARIN OKUTULMASI
       DO 10 J=1,N
   10  READ (5,2) C(J), (A(I,J), I=1,M)
       READ (5,55) (B(I), I=1,M)
       ITE=0
       KX=KX+1
C  TEMEL COZUMDEKI DEGISKENLERI BELIRLEME
       DO 15 J=KX,N
       DO 20 I=1,M
       IF (A(I,J) —1.0) 20,25,20
   20  CONTINUE
       GO TO 15
   25  CI (I) =C(J)
       KCOZUM(I) =J
   15  CONTINUE
C  COZUME GIRECEK DEGISKENLERI BELIRLEME
   75  BUYUK=0.0
       DO 30 J=1,N
       Z(J) =0.0
       DO 35 I=1,M
   35  Z(J) =Z(J) +A(I,J)★CI(I)
       CZF(J) =C(J) —Z(J)
       IF (CZF(J).LE.BUYUK) GO TO 30
       BUYUK=CZF(J)
       IGELEN=J
   30  CONTINUE
```

```
C  CZF POZITIF DEGERLIMIDIR KONTROL ET
       ITE=ITE+1
       IF (BUYUK.LE.0.0) GO TO 40
C  COZUMU TERKEDECEK DEGISKENI BELIRLE
       EKTED=5E25
       DO 45 I=1,M
       IF (A(I, GELEN) ) 45,45,50
   50  TED=B(I)/A(I, IGELEN)
       IF (EKTED.LT.TED) GO TO 45
       EKTED=TED
       IGIDEN=I
   45  CONTINUE
C  PROBLEMIN SINIRLILIGINI BELIRLE
       IF (EKTED.GE.5E25) GO TO 60
C  YENI TABLODA SIRA VE SUTUN DEGISIMI
       KCOZUM (IGIDEN) =IGELEN
       CI (IGIDEN) =C (IGELEN)
       PIVOT=A (IGIDEN, IGELEN)
       B (IGIDEN) =B(IGIDEN) /PIVOT
       DO 65 J=1,N
   65  A(IGIDEN,J) =A(IGIDEN,J) /PIVOT
       DO 70 I=1,M
       IF (I.EQ.IGIDEN) GO TO 70
       SUTUN=A(I, IGELEN)
       B(I) =B(I) —B(IGIDEN) ★SUTUN
       DO 70 J=1,N
       SIRA=A(IGIDEN,J) ★SUTUN
       A(I,J) =A(I,J) —SIRA
   70  CONTINUE
       GO TO 75
C  AMAC FONKSIYONU DEGERINI BELIRLEME
   40  ZMAX=0.0
       DO 80 I=1,M
   80  ZMAX=ZMAX+B(I) ★CI(I)
       WRITE (6,3) ITE, ZMAX
       WRITE (6,4) (KCOZUM (I), B(I), I=1,M)
       GO TO 90
   60  WRITE (6,5)
   90  STOP
       END
```

Yazılmış olan programda simplex tablonun boyutları 10×30 olarak belirtilmiş olmasına karşın bunlar kullanılan bilgisayarın bellek gücüne göre artırılabilir. Programda her sıra ve sütun değişimi sonunda meydana gelen yeni tablo çıktı olarak istenilmemiştir. Ayrıca en son tablodaki katsayılar da çıktı olarak istenilmemektedir. Arzu edilirse ek bir WRITE ifadesi ile bunlar yazdırılabilir. Bu nedenle programda çözüm değişkenleri ve değerlerinin istenilmesi ile yetinilmiştir.

Kuşkusuz daha önce de belirtildiği gibi doğrusal programlama problemlerinin bu programa aracılığı ile çözülebilmesi için ilk temel çözümün belirlenmesi ve eklenecek boş ve yapay değişken sayılarının gerçek değişkenlerle birlikte belirtilmesi gerekmektedir. Diğer yandan programa göre, program girdileri olarak verilecek katsayılar tablodaki sütunlarına göre kartlara aktarılacaktır. Yani, M,N ve KX değişkenlerinin değeri 1. kartta verildikten sonra 2. kartta tablonun katsayıları 11 eleman aynı kartta olmak üzere peş peşe verilecektir. Örneğin;

C(1),A(1,1),A(2,1),A(3,1),.....,A(10,1) ikinci kartta,

C(2),A(1,2),A(2,2),A(3,2),.....,A(10,2) üçüncü kartta v.b.

Katsayılar böylece kartlara delindikten sonra yeni bir kartta B sütunu elemanları aynı biçimde (55 FORMAT'a göre) peş peşe 10 eleman bir kartta olmak üzere kartlara delinecektir.

Böylece başlangıçta verilen ve tablo 1'de gösterilen problemin girdileri kartlara aşağıdaki gibi aktarılacaktır :

1. kart bbbb3bbbb6bbbb3bbbbbbbbbbbbbbb

2. kart bbbbb5bbbbbb10bbbbbbb3bbbbbbb2bb

3. kart bbbbb7bbbbbbb2bbbbbb13bbbbbbb3bb

:

:

7. kart bbbbbbbbbbbbbbbbbbbbbbbbbbbb1bb

8. kart bbb100bbbbb150bbbbb120bb

Yukarıda verilen verilere göre program bilgisayara verildikten sonra program ile birlikte sayfa 182 sonunda verilen çıktı elde edilmiştir.

```
                BURROUGHS LARGE SYSTEMS FORTRAN COMPILATION
                                                        S S F
                                                        = = = =
C   DOGRUSAL PROGRAMLAMA
C   SIMPLEX YONTEMI

      DIMENSION C(30),A(10,30),B(10),CI(10)
      DIMENSION Z(30),CZF(30),KCCZUM(10)
C   FSITLIK VE DEGISKEN SAYILARINI OKUTMA
      READ(5,1)M,N,KX

    1 FORMAT(3I5)
    2 FORMAT(6F8.2)
   55 FORMAT(10F8.2)
    3 FORMAT(1H1,15X,"COZUM",///,
     15X,"TEKRARLAMA SAYISI=",I5,//,
     25X,"AMAC FONKSIYONU DEGERI=",F12.2,///,
     35X,"DEGISKENLER",3X,"DEGERLERI")
    4 FORMAT(5X,I5,10X,F12.2)
    5 FORMAT(15X,"AMAC FONKSIYONU SINIRLI DEGIL")
C   KATSAYILARIN OKUTULMASI
      DO 10 J=1,N
   10 READ(5,2)C(J),(A(I,J),I=1,M)
      READ(5,55) (B(I),I=1,M)
      ITE=0
      KX=KX+1
C   TEMEL COZUMDEKI DEGISKENLERI BELIRLEME
      DO 15 J=KX,N
      DO 20 I=1,M
      IF(A(I,J)-1.0) 20,25,20
   20 CONTINUE
      GO TO 15
   25 CI(I)=C(J)
      KCCZUM(I)=J
   15 CONTINUE
C   COZUMU GIRECEK DEGISKENLERI BELIRLEME
   75 BUYUK=0.0
      DO 30 J=1,N
      Z(J)=0.0
      DO 35 I=1,M
   35 Z(J)=Z(J)+A(I,J)*CI(I)
      CZF(J)=C(J)-Z(J)
      IF(CZF(J).LE.BUYUK)GO TO 30
      BUYUK=CZF(J)
      IGELEN=J
   30 CONTINUE
C   CZF POZITIF DEGERLI MIDIR KONTROL ET
      ITE=ITE+1
      IF(BUYUK.LE.0.0) GO TO 40
C   COZUMU TERKEDECEK DEGISKENI BELIRLE
      EKTED=5E25
      DO 45 I=1,M
      IF(A(I,IGELEN))45,45,50
   50 TED=B(I)/A(I,IGELEN)
      IF(EKTED.LT.TED) GO TO 45
      EKTED=TED
      IGIDEN=I
   45 CONTINUE
C   PROBLEMIN SINIRLILIGINI BELIRLE
      IF(EKTED.GE.5E25) GO TO 60
```

```
C   YENI TABLODA SIRA VE SUTUN DEGISIMI
      KCOZUM(IGIDEN)=IGELEN
      C1(IGIDEN)=C(IGELEN)
      PIVOT=A(IGIDEN,IGELEN)
      B(IGIDEN)=B(IGIDEN)/PIVOT
      DO 65 J=1,M
   65 A(IGIDEN,J)=A(IGIDEN,J)/PIVOT
      DO 70 I=1,M
      IF(I.EQ.IGIDEN) GO TO 70
      SUTUN=A(I,IGELEN)
      B(I)=B(I)-B(IGIDEN)*SUTUN
      DO 70 J=1,N
      SIRA=A(IGIDEN,J)*SUTUN
      A(I,J)=A(I,J)-SIRA
   70 CONTINUE
      GO TO 75
C   AMAC FONKSIYONU DEGERINI BELIRLEME
   40 ZMAX=0.0
      DO 80 I=1,M
   80 ZMAX=ZMAX+B(I)*C1(I)
      WRITE(6,3)ITE,ZMAX

      WRITE(6,4)(KCOZUM(I),B(I),I=1,M)
      GO TO 90



   60 WRITE(6,5)
   90 STOP
      END
```

```
 NO ERRORS DETECTED.  NUMBER OF CARDS =  8.
 COMPILATION TIME = 7 SECONDS ELAPSED,  0.93 SECONDS
 DO STACK SIZE = 7 WORDS.  FILESIZE = 140 WORDS.  ES
 TOTAL PROGRAM CODE = 251 WORDS.  ARRAY STORAGE = 42
 NUMBER OF PROGRAM SEGMENTS = 6.  NUMBER OF DISK SEG
 PROGRAM CODE FILE = (K1709,8)SBF ON METU01.

          COZUM

 TEKRARLAMA SAYISI=     4

 AMAC FONKSIYONU DEGERI=        133.25

 DEGISKENLER   DEGERLERI
      1                  7.79
      2                  7.65
      3                  6.79
```

Not: Diğer programların bilgisayar sonuçlarını bu biçimde klişe verme olanağı çeşitli basım nedenleriyle mümkün olmadı. Ancak bir örnek olarak tüm imkansızlıklara karşın bunu veriyoruz.

Çıktının sonundaki değerler istediğimiz sonuçlardır. Çıktıya göre 4 tablo değişimi (tekrarlama sayısı) gerekmiştir. Amaç fonksiyonunun değeri 133.25 bulunmuştur. Yuvarlama hataları göz önüne alınmazsa, bu değer daha önceki el ile çözümümüzde bulduğumuz $Z_j=133.21$ değeri ile aynıdır. Değişken değerleride aynı biçimde daha önce bulduğumuz değerleri ile aynıdır. Değişkenler altında belirtilen sayılar tablolarımızdaki değişkenlerin dizinlerini belirtmektedir. Yani :

1 (X_1) değeri 7.79

2 (X_2) değeri 7.65

3 (X_3) değeri 6.79 biçiminde anlaşılmalıdır.

EK KAYNAKLAR

1. Aktaş, Ziya ve Epir, Bülent. Elektronik Hesaplayıcılarla Programlama ve Uygulama: Elektronik Hesaplayıcılar ve FORTRAN IV Dili ile Programlama. Ankara: Orta Doğu Teknik Üniversitesi, 1973.

2. Arkun, Erol and Güngör, İsmet. Fundamentals of Programming and FORTRAN IV. Ankara: Middle East Technical University, 1977.

3. Davis, Gordon B. Computer Data Processing. New York: McGraw-Hill Book Company, 1969.

4. Keskinel, Fikret ve Karadoğan, Faruk. FORTRAN IV: Algoritma Kurma ve Program Geliştirme. İkinci Baskı, İstanbul: Üçer Matbaacılık, 1978.

5. McCracken, Daniel D.A Guide to FORTRAN IV Programming. Second Edition. New York: John Wiley and Sons, Inc., 1972.

6. Ralston, Anthony. Introduction to Programming and Computer Science. New York: McGraw-Hill Book Company, 1971.

7. Sturgul, John R. and Merchant, Michael J. Applied FORTRAN IV Programming. Belmont, California: Wadsworth Publishing Company, Inc., 1973.

8. Uman, Nuri. Bilgi İşlemde Kompüterler ve Türkiye'de Kompüterlerin Durumu. Ankara: Siyasal Bilgiler Fakültesi, 1973.

9. Wu, Nesa A. Programming in FORTRAN IV. Dubuque, Iowa: WM.C Brown Company Publishers, 1973.

ISBN 975-482-000-7
A.Ü. S.B.F. ve Basın - Yayın Yüksek Okulu Basımevi, Ankara - 1988 Fiyatı : 1600 TL.