TEMEL YAZILIM KAVRAMLARI

Tarihçe

1642 Yılında *Blaise Pascal* tarafından icat edilen Pascalline Hesap Makinesi, eldeli toplama, ödünç almalı çıkarma yapıyordu.

1671 Yılında *Gottfried Wilhelm Leibniz* tarafından icat edilen Leibniz Çarkı, toplama, ödünç almalı çıkarmanın yanında bölme, çarpma ve karekök işlemlerini yapıyordu.

1801 Yılında *Joseph Maria Jacquard* dokuma tezgahlarında desenleri oluşturmak için delikli kartlar (punch card) kullanmayı icat etmiştir. Sonradan bilgisayara veri girdisi sağlamak ve sonuçların çıktısını göstermek için kullanılacaktır.

1830 Yılında *Charles Babbage* fark makinesi olan analitik makineyi icat etmiştir. Buhar gücü kullanan bu makinenin mantıksal işlem birimi, veri depolama birimi, giriş çıkış üniteleri bulunuyordu. *Augusta Ada Byron*, Babbage ile beraber çalışmıştır. Byron, analitik makinenin bir dizi matematiksel işlemi gerçekleştirebildiğini fark etti ve bu makineyi *Bernoulli* sayılarını üretmek için kullandı. Bu sayıların hesaplanması için yazdığı algoritma tarihteki ilk bilgisayar programı olarak kabul edilir. Bu sebeple *Byron*, ilk programcı olarak kabul görür.

1854 Yılında *George Boole* tarafından ikili sayı sistemini geliştirmiş bugün bilgisayarlarımızın kullandığı bool cebiri üzerine çalışmalar yapmıştır.

1890 Yılında *Herman Hollerith* tarafından delikli kartlarla bilgilerin yüklenebildiği ve bu bilgiler üzerinde toplama işlemlerinin yapılabildiği bir elektro mekanik araç geliştirdi. ABD'nin 1890 nüfus sayımında başarılı biçimde kullanıldı. *Hollerith* başarılı olunca bir şirket kurdu ve daha sonra üç firma işle birleşerek 1924 yılında adını IBM olarak değiştirdi.

1941 Yılında *Konrad Zuze* Z3 isimli elektrik motorları ile çalıştırılan mekanik bir bilgisayar yaptı. Bu (Z1, Z2, Z3 ve Z4 serisi) program kontrollü ilk bilgisayardır.

1944 Yılında *Howard Aitken*, IBM ile iş birliği yaparak MARK I'i yaptı. Bilgiler, MARK I'e delikli kartlarla veriliyor ve sonuçlar yine delikli kartlarla alınıyordu. Saniyede 5 işlem yapabiliyordu. Boyutları,18 m uzunluğunda ve 2,5 m yüksekliğinde idi. İnsan müdahalesi olmadan sürekli olarak, hazırlanan programı yürüten ilk bilgisayar idi ve tam olarak elektronik bir bilgisayar değildi.

1943-1945 Yılları arasında *Konrad Zuze*, ilk yüksek düzey programlama dili olan Plankalkül'ü, Z1 bilgisayarı için geliştirmiştir. Emir kodlarını (instruction code) ve sembolik isimleri (mnemonic) içermeyen programlama dilleri yüksek düzey programlama dili (high-level programming language) olarak adlandırılır. Yüksek düzey programlama dilleri çoğunlukla İngilizce sözcükleri kullanır.

1946 Yılında *John Von Neumann* önderliğinde Pensilvanya Üniversitesinde ENIAC (Elektronik Sayısal Hesaplayıcı ve Doğrulayıcı) isimli sayısal elektronik bilgisayar tamamlandı. Askeri amaçla üretildi ve top mermilerinin menzillerini hesaplamak için kullanıldı. *Neumann* tarafından geliştirilmiş bu mimari (Stored-program computer and Universal Turing machine § Stored-program computer), günümüzdeki bilgisayarlarda hala kullanılmaktadır. 18,000 adet elektronik tüp kullanılan ENIAC; 150 kW gücünde idi ve 50 ton ağırlığıyla 167 m^2 yer kaplıyordu. Saniyede 5.000 toplama işlemi yapabiliyordu. Mark-I'den 1000 kat daha hızlıydı.

Bir önceki bölümde anlatıldığı üzere emir kodu veya sembolik isimlerle yazılan programlar düşük düzey programlama (low-level programming) olarak adlandırılır. Bu programlamanın yapılabilmesi için hem işlemcinin hem de hafıza ve işlemciye bağlanan diğer bileşenlerin nasıl çalıştığı ve tasarlandığını çok iyi bilinmesi gerekir. Daha çok elektronik ve bilgisayar mühendisleri ya da sistemi tasarlayan matematikçiler bu programlama türünü kullanır.

Genel Amaçlı Yüksek Düzey Diller

Sembolik isimler yerine bildiğimiz konuşma diline yakın talimatlar olarak metin olarak yazılan programlara ilişkin diller, yüksek düzey programlama dilleri (high-level programming language) olarak adlandırılır. Yani emir kodu (instruction code) ve sembolik isim (mnemonic) kullanmayan programlama dilleri yüksek düzey programlama dileridir.

Yüksek düzey dillerde talimatlar (statements) olarak yazılan programlar yine kendine has derleyiciler tarafından makine diline çevrilir. Her talimat, genellikle birden fazla makine kodu ya da emirden (instruction) oluşur.

1954 Yılında FORTRAN (FORmula TRANslation), *John Backus* liderliğinde IBM 704 için geliştirilirdi. 32 farklı talimata (statement) sahiptir. İlk genel amaçlı, yüksek düzey bir programlama dilidir.

Yüksek seviyeli bir dilde yazılmış bir talimat (statement), bilgisayara belirtilen bir eylemi gerçekleştirmesini söyler. Yüksek seviyeli bir dildeki tek bir talimat, birkaç emri (instruction) içerebilir. Talimatlar, yapılaması gerekeni kısa ve net olarak belirten mantıksal cümlelerdir (logical sequences).

Aşağıda üç kenarı teyp ünitesinden alınan bir üçgenin alanını hesaplayan FORTRAN II programı verilmiştir[5]. Görüldüğü üzere talimatlar kısa öz ve anlaşılırdır.

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT -
C INTEGER VARIABLES START WITH I,J,K,L,M OR N
      READ(5,501) IA,IB,IC
  501 FORMAT(3I5)
      IF (IA) 701, 777, 701
  701 IF (IB) 702, 777, 702
  702 IF (IC) 703, 777, 703
  777 STOP 1
  703 S = (IA + IB + IC) / 2.0
      AREA = SQRT( S * (S - IA) * (S - IB) * (S - IC) )
      WRITE(6,801) IA,IB,IC,AREA
  801 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
     $13H SQUARE UNITS)
      STOP
      END
```

1956 Yılında ticari amaçlarla kullanılabilen ve seri halde üretimi yapılan ilk bilgisayar UNIVAC I oldu. UNIVAC, giriş-çıkış birimleri manyetik bant idi ve bir yazıcıya sahipti. UNIVAC 1951-1959 arasında üretilen bilgisayarlarda vakum tüpleri kullanıldı. Bu tüpler bir ampul büyüklüğünde, çok fazla enerji harcamakta ve çok fazla ısı yaymakta idiler. Veri ve programlar manyetik teyp ve tambur gibi bilgi saklama araçlarıyla saklandı. Veriler ve programlar bilgisayara delgi kartları ile yükleniyordu. 1959 yılı sonrasında üretilen bilgisayarlarda transistörler (yaklaşık 10 bin adet) kullanıldı. Artık transistor içeren ikinci kuşak bilgisayarlar hayatımıza girmiştir.

Arapsaçı Kod

1955-1959 Yılları arasında FLOW-MATIC:B0 (Business Language Version 0) Dili, *Grace Hopper* tarafından UNIVAC I için geliştirildi. İngilizceye benzeyen ilk yüksek düzey programlama dilidir. Aşağıda B0 Dilinde örnek bir program verilmiştir[6].

```
 (0) INPUT INVENTORY FILE-A PRICE FILE-B ; OUTPUT PRICED-INV FILE-C UNPRICED-INV
     FILE-D ; HSP D .
 (1)  COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION 10 ;
     IF EQUAL GO TO OPERATION 5 ; OTHERWISE GO TO OPERATION 2 .
```

[5] https://github.com/scivision/fortran-II-examples/blob/main/herons_formula.f

[6] https://gist.github.com/marcgg/f9f2da31a973ba319809

```
 (2)  TRANSFER A TO D .
 (3)  WRITE-ITEM D .
 (4)  JUMP TO OPERATION 8 .
 (5)  TRANSFER A TO C .
 (6)  MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
 (7)  WRITE-ITEM C .
 (8)  READ-ITEM A ; IF END OF DATA GO TO OPERATION 14 .
 (9)  JUMP TO OPERATION 1 .
(10)  READ-ITEM B ; IF END OF DATA GO TO OPERATION 12 .
(11)  JUMP TO OPERATION 1 .
(12)  SET OPERATION 9 TO GO TO OPERATION 2 .
(13)  JUMP TO OPERATION 2 .
(14)  TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ ; IF EQUAL GO TO OPERATION 16 ;
     OTHERWISE GO TO OPERATION 15 .
(15)  REWIND B .
(16)  CLOSE-OUT FILES C ; D .
(17)  STOP . (END)
```

Görüldüğü üzere bu tarihlere kadar yazılan programlarda bir sürü GO TO ya da JUMP TO talimatı (statement) bulunmaktadır. Bu durumda programın ne yaptığı konusunda fikir yürütmeye çalıştığımızda okunaklılık oldukça azdır. İşte okunaklılığın az olduğu bu program kodlarına arapsaçı kod (spaghetti code) adı verilir. Günümüzde ne yaptığı anlaşılamayan kodlar için bu kavram çokça kullanılmaktadır.

1958 Yılında LISP programlama dili, *John McCarthy* önderliğinde *Steve Russell* tarafından geliştirilmiş ikinci yüksek düzey programlama dilidir.

1959 Yılında Common Business-Oriented Language-COBOL programlama dili, *CODASYL* komitesi tarafından İngilizceye benzer ikinci yüksek düzey bir dil olarak geliştirilmiştir.

1958 Yılında daha sonra ALGOritmic Language-ALGOL adını alan International Algebraic Language-IAL Programlama Dili geliştirilmiştir. Bu dil daha sonra ortaya çıkan birçok dile önderlik etmiştir. Bu dil ile birlikte kod blokları (code block), iç içe fonksiyon (nested function), değişken kapsamı (scope) ve dizi (array) kavramları programcının hayatına girmiştir. ALGOL dili, barındırdığı özellikler sebebiyle, kendisinden sonra ortaya çıkan bütün programlama dilleri için bir esin kaynağı olmuştur.

İşin doğası gereği makine diline çevrilecek tüm metinler gözle okunabilmektedir. Yani bütün kaynak kodlar gözle okunabilir.

Değişken

Değişkenler (variable) aslında cebirden bildiğimiz değişkenlerdir. Cebir (algebra) işlemlerinde doğrudan problemde verilen sayıları değil, onları temsil eden değişkenleri kullanırız.

$$3x^2 + 2x + 10$$

$$c^2 = a^2 + b^2$$

$$c = 2\pi r$$

Yukarıdaki cebirsel ifadelerin ilkinde bir polinom, ikincisinde ise bir dik üçgenin kenar uzunlukları arasındaki ilişki verilmiştir. Bu ifadelerde `x`, `a`, `b` ve `r` bağımsız değişken, `c` ise bağımlı değişkendir. Aslında problemi çözerken bunlar sayılara karşılık gelir. Örneklerdeki `3`,`2`,`10`,`2` ve `π` ise değişmez (literal) sayılardır.

Bilgisayarda ise sayıları tutacak değişkenler, bir miktar bellek bölgesini işgal eder ve bu bellek bölgesinde çeşitli biçimlerde tutulurlar. Tamsayılar, ikili (binary) sayı olarak bellekte tutulur. Bu ikili sayının en anlamlı bitinin işaret biti olarak kullanılıp kullanılmayacağına göre bellekte biçimlendirilir;

Bit Sayısı	Bellek Miktarı	Sayı Sınırları
8 Bit	bayt	-127 ile +128 arasında veya 0 ile 255 arasında

16 bit	2 bayt	-32.767 ile +32.768 arasında veya 0 ile 65.535 arasında
32 bit	4 bayt	-2.147.483.648 ile +2.147.483.647 veya 0 ile 4.294.967.295 arasında
64 bit	8 bayt	-9.223.372.036.854.775.808 ile +9.223.372.036.854.775.807 arasında veya 0 ile 18,446,744,073,709,551,615 arsında

Tablo 3. Tamsayı Değişkenler ve Sayı Sınırları

Gerçek sayılar kayan noktalı sayı (floating-point number) olarak biçimlendirilir; 1914 yılında *Leonardo Torres Quevedo* tarafından Charles Babage'ın analitik makinesine dayalı kayan nokta analizi yayınladı.

$$12345 = 1234{,}5x10^1 = 123{,}45x10^2 = 12{,}345x10^3 = 1{,}2345x10^4 = 0{,}12345x10^5$$

Yukarıda ondalık tabanda verilen gerçek bir sayıya ilişkin kesir noktasının kaydırılması ile aynı sayı ifade edildiği bir örnek verilmiştir. Kesir noktası sola kaydırıldığında 10 ile çarpılmış, sağa kaydırıldığında ise 10 ile bölünmüş olur. Kayan noktalı sayının üs kısmı dışında kalan kısmı taban (base) veya kesir (fraction) olarak adlandırılır. Tabanın kesir noktası sağında kalan hanelerine ise mantis (mantissa) adı verilir.

1938 yılında *Konrad Zuse*, ilk ikili (binary) programlanabilir mekanik bilgisayar olan Z1'i yapmıştı. Bu bilgisayar, 7 bitlik işaretli üs, 17 bitlik anlamlı değer ve bir işaret biti içeren 24 bitlik ikili tabanda kayan noktalı sayı kullanmıştır.

Bilgisayarlarda tamsayılarda olduğu gibi kayan noktalı sayılar da IEEE-754 standardında ikili sayı tabanında tutulur[7]. Örneğin ondalık tabanda 50,25 sayısı ikili tabanda 1.1001001 x2^5 olarak ifade edilir.

Bit Sayısı	Kapladığı Bellek Miktarı	Sayı Sınırları
32 bit	4 bayt	$1{,}2x10^{-38}$ ... $3{,}4x10^{+38}$ Tek hassasiyetli gerçek sayılar; en soldaki 1 bit işaret biti, sonraki 8 bit üs ve geri kalan 23 bit ise kesir olarak kullanılan ikilik tabanda saylardır.
64 bit	8 bayt	$2{,}3x10^{-308}$ ila $1{,}7x10^{+308}$ Çift hassasiyetli gerçek sayılar; en soldaki 1 bit işaret biti, sonraki 11 bit üs ve geri kalan 52 bit ise kesir olarak kullanılan ikilik tabanda saylardır.

Tablo 4. Kayan Noktalı Sayılar ve Sınırları

Program yazılırken tanımlanacak değişkenlere ilişkin bellek ihtiyacı programcı tarafından belirlenir. Programcı değişkene tahsis edilecek bellek miktarına ve tutacağı içeriğe göre veri tipi (data type) belirler. Programcı tarafından kullanılacağı amaca göre belirleyeceği veri tipinde istediği kadar değişken tanımlayabilir.

Fonksiyon

Matematikte fonksiyon (function), bir veya daha fazla kümenin elemanını tek bir kümenin elemanına eşleyen ilişkidir (relationship). Girdi olan bağımsız değişkenler ile çıktı olan bağımlı değişken arasındaki bu ilişki, matematiksel olarak aşağıdaki şekilde ifade (expression) edilir.

Girdi	İlişki	Çıktı
1	x2	2
2	x2	4
3	x2	6
4	x2	8
6	x2	12
...		

Tablo 5. İki ile çarpma fonksiyonu.

$$f(x) = 2x$$

[7] IEEE Computer Society (2019-07-22). IEEE Standard for Floating-Point Arithmetic. IEEE STD 754-2019. IEEE. pp. 1–84.

Burada f, fonksiyon anlamında x ise girdi anlamında olup parametre olarak adlandırılır. Böylece tabloda verilen fonksiyon tablosuna her girdiyi yazmak zorunda kalmayız. Bazen fonksiyona aşağıdaki örneği verilen şekilde bir ad verilmez;

$$y = 2x$$

Bazı durumlarda ise girdi daha detaylı tanımlanır;

$$\mathbf{x \in R} \text{ olmak üzere } \mathbf{f(x)=2x+3}$$

Bu fonksiyon, reel sayı kümesindeki her `x` elemanını, hesaplanan `f(x)` değerine eşleyen bir ifadedir. Yani `x=1.0` argümanını `f(1.0)=5.0` reel sayısına eşler. Kısaca fonksiyon `5.0` reel sayısını hesaplayıp döndürür (return).

$$\mathbf{a \in R} \text{ ve } \boldsymbol{b \in Z} \text{ olmak üzere } \mathbf{g(a,}\boldsymbol{b}\mathbf{)=2a+b}$$

Bu fonksiyon ise reel sayı kümesindeki her `a` elemanı ile tamsayı kümesindeki `b` elemanını, hesaplanan `g(a,b)` değerine eşleyen bir ifadedir. Yani `a=1.0` ve `b=1` argümanlarını `g(1.0,1)` fonksiyonu yine reel sayı kümesinden `3.0` reel sayısına eşler. Kısaca fonksiyon, `3.0` reel sayısını hesaplayıp döndürür (return). Buradaki `x`, `a` ve `b` parametre olarak adlandırılır. `f(1.0)` ifadesindeki `1.0` ise parametrenin o an andığı değeri belirtir ve gerçek parametre (actual parameter) ya da argüman (argument) olarak adlandırılır. Bu fonksiyonlar, `f(3.0)+g(3.0,2)+f(2.0)+g(1.0,3)` gibi tanımlandıktan sonra tekrar tekrar çağrılabilirler (call).

Yapısal Programlama

1958 Yılındaki FORTRAN-II diline çıktı üretmeyen fonksiyonlar için SUBROUTINE, değer üreten fonksiyonlar için FUNCTION, CALL ve RETURN özellikleri eklenmiştir.

1958 ile 1969 Yılları arasında ALGOL diline çıktı üretmeyen fonksiyonlar için PROCEDURE ve BEGIN-END arasında kod bloğu özellikleri eklenmiştir.

1966 Yılındaki FORTRAN-IV diline INTEGER, REAL, DOUBLE PRECISION, COMPLEX ve LOGICAL veri tipleri (data type) ile Mantıksal IF, aritmetik (üç yollu) IF ve DO döngüsü talimatları (statement) eklenmiştir.

Bu yıllarda her ne kadar çıktı üretmeyen fonksiyonlar için SUBROUTINE, birden fazla çağrı noktası olabilen COROUTINE ve FUNCTION kullanılıyor olmasına rağmen hala GO TO/JUMP TO ifadeleri kullanılmakta ve kafa karışıklığına devam etmekteydi. 1968 Yılında *Edsger W. Dijkstra* önderliğinde GO TO/JUMP TO ifadesini zararlı olarak ilan edilmiştir.

1970 Yılında *Niklaus Wirth* tarafından Pascal dili geliştirildi. Bu dil ilk geliştirilen yapısal programlama dilidir. Yapısal programlamada (structural programming) verileri taşıyan veri yapıları (değişkenler ve diziler) ile bunu işleyen kontrol yapıları (if, for, do, repeat) birbirinden ayrılmıştır. Pascal dili, kendi kendini çağıran özyinelemeli (recursive) fonksiyonlar ile değişkenlerin adreslerini tutan göstericileri (pointer) desteklemektedir[8].

Yapısal programlamanın genel çerçevesi;

1. Programın başladığı yeri belirten bir ana fonksiyon (main function) tanımlanır. Kodlaması yapılacak işi birbirini çağıran fonksiyonlar yazarak şeklinde burada yazarız. Yani programlama genel olarak fonksiyonların birbirini çağırmasıyla yapılır.
2. Her fonksiyonda ilk önce veri yapıları (data structure) tanımlanır. Veri yapıları;
   a. En basit veri yapısı `char`, `integer`, `float` ve `double` gibi ilkel veri tiplerinden (primitive data type) olabilen değişken (variable) olabileceği gibi,
   b. İlkel veri tiplerinden meydana gelen dizi (array),
   c. İlkel veri tiplerinin birkaçından veya dizilerinden bir araya gelmesiyle oluşan yapılar (structure),

[8] Wirth, Niklaus (1976). Algorithms + Data Structures = Programs. Prentice-Hall. p. 126

   d. Ya da dinamik olarak yani çalıştırma anında birbirine bağlanmış verilerden oluşan bağlı liste (linked list) olabilir.
3. Her fonksiyonda, tanımlanan veri yapılarının ardından, bu yapıları işleyecek kontrol yapıları (control structure) tanımlanır. Kontrol yapıları bir veya daha fazla veya iç içe `if`, `if-else`, `switch`, `do`, `while`, `for`, `continue`, `break`, `goto` ve `return` talimatlarından (statement) oluşur.

Pascal dilinde ana fonksiyon nokta ile biten BEGIN-END bloğudur. C ve C++ dillerinde ana fonksiyon, `main()` fonksiyonudur.

C Programlama Dili, *Dennis M. Ritchie* tarafından Bell Laboratuvarlarında UNIX işletim sistemini geliştirmek için geliştirilen genel amaçlı, üst düzey bir dildir. C, ilk olarak 1972'de DEC PDP-11 bilgisayarında çalıştırılmıştır. *Ken Thompson* tarafından geliştirilen B adlı daha eski bir dilden gelişmiştir.

1978 yılında *Brian Kernighan* ve *Dennis Ritchie*, günümüzde K&R standardı olarak bilinen C'nin ilk kamuya açık kılavuzunu hazırlamışlardır. C programlama dili;

– Öğrenmesi kolaydır,
– Yapısal (structural) programlama dilidir,
– Hızlı ve verimli (efficient) programlar üretir,
– Assembly dili desteği ile düşük düzey programlamayı (low-level programming) da destekler,
– Standart başlık dosyaları sayesinde çeşitli bilgisayar platformlarında derlenebilir.

Bu özellikleri dolayısıyla neredeyse tüm işletim sistemlerinin çekirdeği C dilinde yazılmıştır ve sistem dili olarak da adlandırılmaktadır.

Yapısal programlamada veri ile bunu işleyen yapılar birbirinden ayrıldığından arapsaçı programlamanın aksine okunaklılık oldukça yüksektir.

Nesne Yönelimli Programlama İhtiyacı

C++ programlama dili, C diline yapılan nesne yönelimli programlama (object oriented programming-OOP) eklentileri yapılarak geliştirilmiş bir programlama dilidir. 1979 senesinde bir Danimarkalı bilgisayar bilim adamı olan *Bjarne Stroustrup*, sonradan C++ olarak bilinecek olan "C with Classes" üzerinde çalışmaya başladı ve 1985 yılında ilk sürümünü yayınladı.

Yapısal programlamada talimatlar (statements) art arda koda yazılarak programlama yapılır ve programların neler yaptığı bu talimatlar izlenerek anlaşılabilir. Talimatların zincirin halkaları gibi birbirinin peşi sıra yazılarak yapılan programlamaya emreden programlama (imperative programming) paradigması adı verilir. Bu paradigmaya sahip diller, yazılımı yapılacak sürece ilişkin nelerin yapılacağını değil, işin nasıl yapılacağını belirtirler. Emreden paradigmanın bir örneği olan yapısal programlamada;

– Kod ne kadar büyürse, modüllere ayırma imkânı olmasına rağmen, fonksiyonlar arasındaki bağımlılık da o kadar artar. Bir fonksiyonun parametrelerindeki değişiklik, onun kullanıldığı tüm yerlerde değişiklik gerektirir. Değişim yönetimi (change management) çok zordur ve uzun zaman alır.
– Hata durumunda yapılacak işlemlere ilişkin kod ile iş sürecini gerçekleştiren kod iç içedir. Aynı fonksiyonun bazı durumlarda hata, bazı durumlarda ise değer döndürmesi mümkündür. Çoğu durumda hataların izini sürmek (error handling) işi zorlaşır.
– Yapısal programlamada, gösterici (pointer) kullanımında erişilmesi istenmeyen bellek bölgelerine erişilmesi halinde istenmeyen program davranışları ortaya çıkar. Kodun güvenli (safe code) olarak çalıştığının incelenmesi çok zordur.
– Sürekli olarak benzer projelerde aynı kodları tekrar yazmak (duplicate code) gereklidir.
– Emreden paradigmanın burada belirtilen sebepler başta olmak üzere çeşitli problemleri bulunmaktadır. Bu nedenle 1980'li yıllarda birçok yazılım projesi başarısız (fail) olmuştur. Bir yazılım; Kullanıcı ihtiyaçlarının karşılanamaması, Öngörülen bütçenin aşılması ve Zamanında teslim edilememesi durumlarında başarısız olur;

Yapısal programlamadaki bu sıkıntıları gören yazılımcıların imdadına Simula ve Smalltalk programlama dillindeki yaklaşım yetişmiştir. Bu diller oldukça yavaş olmasına rağmen getirdiği çözümler oldukça yenilikçiydi.

Smalltalk, 1972 yılında Xerox Park şirketinde *Alan Kay* önderliğinde *Dan Ingalls*, *Adele Goldberg*, *Ted Kaehler*, *Diana Merry*, *Scott Wallace* ve *Peter Deutsch* tarafından geliştirilmiş bir dildir. Smalltalk dilinde nesneler (object) temel yapıtaşlarıdır. Nesneler;

- Durum (state) ve davranışlara (behavior) sahiptir.
- Programlama, nesnelerin birbirlerine ileti göndermesiyle (message-passing) yapılır.
- Nesnelerin kendi ya da bir başka nesnenin davranış ve durumlarını öğrenebilme yeteneği yani yansıma (reflection) özelliği vardır.

Emreden paradigma (imprative programming) aksine nesnelerin birbirlerine ileti göndermesi bakış açısıyla yapılan programlamaya nesne yönelimli programlama (object oriented programming) paradigması adı verilir.