



BMB214 Programlama Dilleri Prensipleri

Ders 2. Başlıca Programlama Dillerinin Gelişimi

Konular

- ⊙ Alan Turing vs. Alonzo Church
- ⊙ Zuse's Plankalkül
- ⊙ Minimal Donanım Programlama: Pseudocodes
- ⊙ Fortran
- ⊙ Fonksiyonel Programlama: Lisp
- ⊙ Gelişmişliğe Doğru İlk Adım: ALGOL 60
- ⊙ İşletme Kayıtlarını Bilgisayarlaştırma: COBOL
- ⊙ Zaman Paylaşımının Başlangıcı: Basic
- ⊙ Herkes İçin Her Şey: PL / I
- ⊙ İki Erken Dinamik Dil: APL ve SNOBOL
- ⊙ Veri Soyutlamanın Başlangıcı: SIMULA 67

Konular (...)

- ⊙ Ortogonal Tasarım: ALGOL 68
 - ALGOL'ün Torunları Pascal ve C
- ⊙ Mantığa Dayalı Programlama: Prolog
- ⊙ Tarihin En Büyük Tasarım Çabası: Ada
- ⊙ Nesneye Yönelik Programlama: Smalltalk
- ⊙ Emir Esaslı ve Nesnesine Yönelik Özellikleri Birleştirme: C ++
- ⊙ Objective C, Eiffel, Delphi
- ⊙ Emir Esaslı Nesneye Yönelik Bir Dil: Java
- ⊙ Betik Dilleri
- ⊙ .NET'in Amiral Gemisi Dili: C #
- ⊙ Performansı Ön Plana Çıkaran Dil: Rust
- ⊙ İstatistiksel hesaplama: R Programlama Dili
- ⊙ Fonksiyonel Programlamada Yeni Trend: Haskell
- ⊙ Diğer diller

Alan Turing vs. Alonzo Church

Alan Turing



"On Computable Numbers" 1937

Alonzo Church



"A set of postulates for the foundation of logic", 1932

Alan Turing vs. Alonzo Church

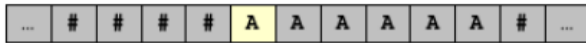
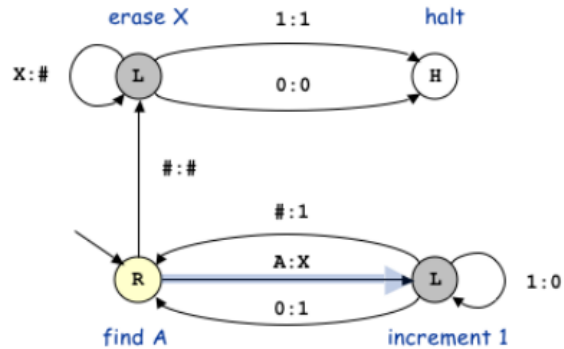
- © Turing makinesi denilen algoritma tanımı ile modern bilgisayarların kavramsal temelini attı.
- © Princeton'da beraber çalıştığı tez hocası **Alonzo Church** ile geliştirdiği Church-Turing Hipotezi ile de matematik tarihine geçmiştir. Bu tez, bir algoritmayla tarif edilebilecek tüm hesaplamaların dört işlem, projeksiyon, ekleme ve tarama operasyonları ile tarif edilebilecek hesaplamalardan ibaret olduğunu ifade eder.
- © Turing Makinasını, herhangi bir şeyin “hesaplanabilir” olup olmadığı gösterebilmek için geliştirdi.
- © Bir sistem basit Turing makinası emirleri ile tanımlanabiliyorsa bu işlem “Turing Bütünlüğü (Turing Completeness)” içeriyor anlamına gelir.
- © Lamda kalkülüs (λ -calculus), herhangi bir tek bantlı Turing makinesini simule edebilen evrensel bir hesaplama modelidir. Soyutlama ve işlev çağırmaya dayanmaktadır. Alonzo Church tarafından 1930'larda matematiğin temelleri üzerine bir araştırma olarak ortaya koyulmuştur.

https://tr.wikipedia.org/wiki/Alan_Turing

https://tr.wikipedia.org/wiki/Lamda_kalk%C3%BCI%C3%BCs

Alan Turing vs. Alonzo Church

Turing machines



Lambda calculus

$$\begin{aligned} [x \rightarrow y] x &= y \\ [x \rightarrow y] z &= z \\ [x \rightarrow y] \lambda z . x &= \lambda z . x \\ [x \rightarrow y] \lambda y . x y &= \lambda y' . y y' \\ [x \rightarrow y] x (\lambda x . x) &= y (\lambda x . x) \end{aligned}$$

Alan Turing vs. Alonzo Church

Turing languages

1957 – FORTRAN
1959 – COBOL, ALGOL
1962 – SIMULA

1972 – C, Smalltalk

1979 – C++

1991 – Python

1995 – Java

Church languages

1959 – LISP

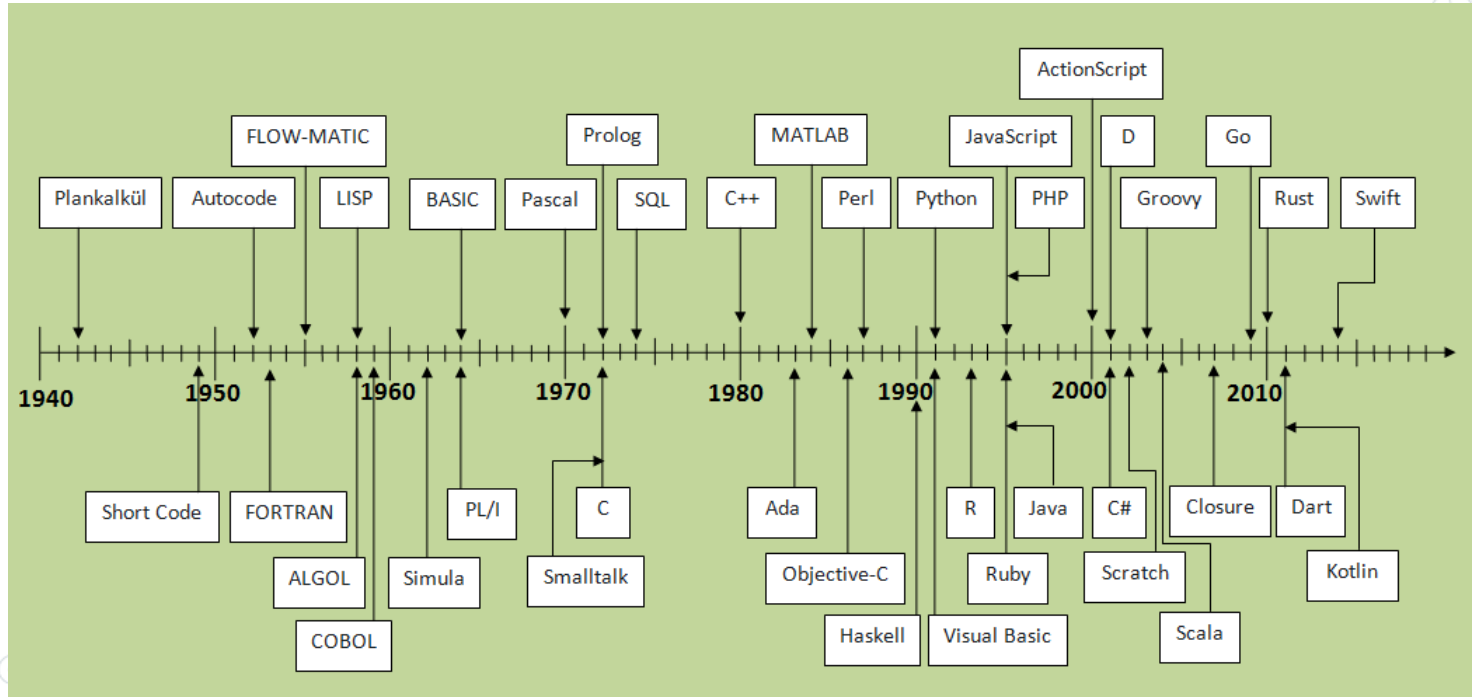
1966 – ISWIM

1972 – Prolog

1978 – ML

1990 – Haskell

Başlıca Programlama Dillerinin Zaman Çizelgesi



http://cdn.oreillystatic.com/news/graphics/prog_lang_poster.pdf

Bir makine kodu ve Assembly Dili örneği

```
b8 21 0a 00 00 #moving "!\\n" into eax
a3 0c 10 00 06 #moving eax into first memory location
b8 6f 72 6c 64 #moving "orld" into eax
a3 08 10 00 06 #moving eax into next memory location
b8 6f 2c 20 57 #moving "o, W" into eax
a3 04 10 00 06 #moving eax into next memory location
b8 48 65 6c 6c #moving "Hell" into eax
a3 00 10 00 06 #moving eax into next memory location
b9 00 10 00 06 #moving pointer to start of memory location into ecx
ba 10 00 00 00 #moving string size into edx
bb 01 00 00 00 #moving "stdout" number to ebx
b8 04 00 00 00 #moving "print out" syscall number to eax
cd 80          #calling the linux kernel to execute our print to stdout
b8 01 00 00 00 #moving "sys_exit" call number to eax
cd 80          #executing it via linux sys_call
```

```
global _main
extern _printf

section .text

_main:
    push    message
    call    _printf
    add     esp, 4
    ret

message:
    db 'Hello, World', 10, 0
```

Zuse's Plankalkül

- ◎ Bir bilgisayar için tasarlanmış ilk yüksek seviye programlama diliydi.
- ◎ 1945'te tasarlandı
- ◎ İlk derleyici, Joachim Hohmann tarafından 1975'teki tezinde uygulandı.
- ◎ Temel Özellikleri
 - En basit veri Türü tek bittir.
 - C'de struct'lara benzer arrays ve records tipleri vardı.
 - Sadece yerel değişkenler.
 - İşlevler özyinelemeyi desteklemiyor
 - Yalnızca değere göre aramayı destekler
 - Koşullu ifadeler içerir
 - for döngüsü benzer fin komutu var
 - Goto komutu Yok.
- ◎ Birçok programlama dilinin (Cobol, FLOW-MATIC, LISP, AutoCode) geliştirilmesine öncülük etti.

Zuse's Plankalkül

© A [4] + 1 ifadesini A [5]'e atamak için

		A + 1 => A		
V		4	5	(subscripts)
S		1.n	1.n	(data types)

Temel veri tipi olarak S0 ile temsil edilen bir bitti.

8 bitlik bir veri tipi 8 * S0

n bitlik bir veri tipi n * S0 ve S1.n şeklinde ifade edilir.

Minimal Donanım Programlama: (1940 sonları, 1950 başları)

Pseudocodes

- ◎ Makine kodunun zorlukları
 - Yorumlamalı
 - Zayıf okunabilirlik
 - Zayıf değiştirilebilirlik
 - İfade kodlaması sıkıcı ve zor
 - Eksikliği- indeksleme veya kayan nokta yok
- ◎ Kaba kod ile karıştırmayın, böyle bir programlama dili vardır.

Pseudocodes - Short Code

- ◎ Mauchly tarafından 1949'da BINAC bilgisayarlar için geliştirildi.
- ◎ Makine koduna göre 50 kat yavaş. (Yorumlama işlemi var)
- ◎ İfadeler soldan sağa kodlandı.
- ◎ Örnek
 - $a = (b + c) / b * c$
 - $X3 = (X1 + Y1) / X1 * Y1$ ikame değişkenleri
 - X3 03 09 X1 07 Y1 02 04 X1 Y1 (operatör ve parantezler temsil edilir.)
 - Yukarıdaki ifade 12 baytlık olarak ayrılır.
 - 07Y10204X1Y1
 - 0000X30309X1

Pseudocodes - Speedcoding

- ◎ 1954'te IBM 701 için Backus tarafından geliştirildi.
- ◎ Temel Özellikleri
 - Doğal Dil ifadeleri içerir
 - Aritmetik (4 işlem) ve matematik fonksiyonlar (square root, sine, arc tangent, exponent ve logarithm) için pseudo kodlar
 - Koşullu ve koşulsuz dallanma
 - Dizi erişimi için otomatik artış kayıtları
 - Yavaş! (Yorumlayıcı var)
 - Kullanıcı programı için sadece 700 kelime kaldı

Pseudocodes - Diğer Sistemler

◎ UNIVAC Derleme Sistemi (Compiling System)

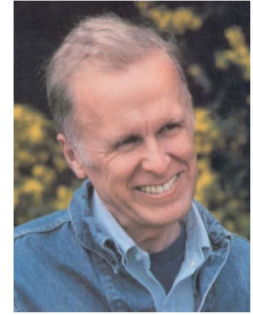
- Grace Hopper liderliğindeki bir ekip tarafından geliştirildi
- Pseudocode, makine koduna genişletildi. (Bu sayede kaynak kod kısaldı.)

◎ David J. Wheeler (Cambridge Üniversitesi)

- Mutlak adresleme problemini çözmek için yeniden konumlandırılabilir adres bloklarını kullanma yöntemi geliştirdi. (Bellek dinamik kullanılacak.)

IBM 704 ve Fortran

```
program helloworld  
  print *, "Hello world!"  
end program helloworld
```



John Backus,
"History of FORTRAN" 1978

- ◎ **FORM**ula **TRAN**slating System: Fortran
- ◎ Fortran 0: 1954 - uygulanmadı
- ◎ John Backus ve ekibi tarafından geliştirildi.
- ◎ Fortran I: 1957
 - Dizin kayıtları ve kayan nokta donanımı olan yeni IBM 704 için tasarlandı
 - Derlenmiş programlama dilleri fikrine yol açtı, çünkü yorumlama işlemi maliyetli idi.
 - Derleyicisi olan bir dil
 - Derleme sayesinde makine kodu ile yazılmış kod gibi hızlı çalışan bir yapı hedeflendi.
 - Geliştirme ortamı
 - Bilgisayarlar küçüktü ve güvenilirmezdi
 - Uygulamalar bilimseldi
 - Programlama metodolojisi veya aracı yok
 - Makine verimliliği en önemli endişeydi

<https://tr.wikipedia.org/wiki/Fortran>
<https://en.wikipedia.org/wiki/Fortran>

Fortran I

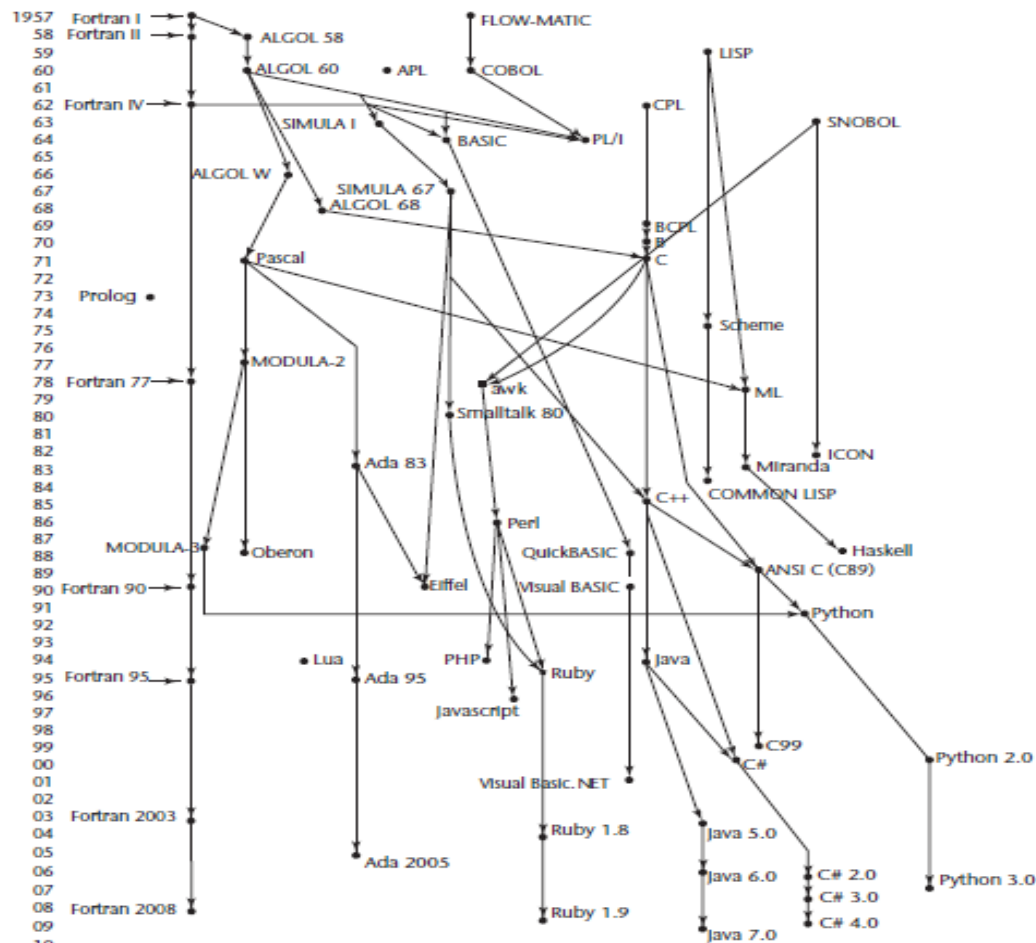
- ⊙ İsimler altı karaktere kadar olabilir
- ⊙ iterative (loop) **Do** ifadesi
 - ***Do N1 variable = first_value, last_value***
- ⊙ Biçimlendirilmiş G / Ç (Formatted I/O): FORMAT, READ, READ INPUT TAPE, WRITE, WRITE OUTPUT TAPE, PRINT, and PUNCH
- ⊙ Biçimlendirilmemiş G / Ç (Unformatted I/O): READ TAPE, READ DRUM, WRITE TAPE, and WRITE DRUM
- ⊙ Kullanıcı tanımlı alt programlar
- ⊙ Üç dala sahip IF ifadesi (aritmetik IF)
 - ***If (arithmetic_expression) negative, zero, or positive***
- ⊙ Çalışma sırasında veri tiplendirme ifadeleri ve bellek tahsisi yoktu
- ⊙ Matris, denklem sistemleri ve diff. Denklem sistemlerini kolaylaştırıyordu

Fortran I ...

◎ FORTRAN'ın ilk uygulanan versiyonu

- Ayrı bir derleme yok
- Derleyici, 18 işçi bir yıllık çalışması ile Nisan 1957'de yayınlandı
- 400 satırdan büyük programlar, esasen 704'ün düşük güvenilirliği nedeniyle nadiren doğru şekilde derlenir
- Kod çok hızlıydı
- Yaygın olarak kullanıldı
- Yeni programlama dillerine ilham kaynağı oldu
- Çalışma anında yeni değişken yaratılmıyordu. Başka bir deyişle bellek tekrar düzenlenemiyordu.

Evolution of the Major Programming Languages



Fortran II

- ◎ 1958'de dağıtıldı
 - Bağımsız derleme: Bağımsız derleme olmadan önde ufak bir kod değişikliğinde dahi tüm kod derleniyordu.
 - Getirdiği en önemli yenilik alt-programların ayrı ayrı derlenebilmesini sağlamaktı. Böylece alt-programda yapılan küçük bir değişim için tüm programı yeniden derlemek yerine, sadece alt-programın derlenmesi sağlandı. Bu zor olan ve genellikle makine hatası sonucu yarıda kalan derleme işlemine büyük bir kolaylık sağladı.
 - Hatalar düzeltildi
 - Gelen bazı ifadeler
 - SUBROUTINE, FUNCTION, and END
 - CALL and RETURN
 - COMMON

Fortran IV

- ◎ Fortran III ürüne dönüşmedi.
- ◎ *Fortran IV* zamanın en geniş alanda kullanılmış programlama dili oldu.
- ◎ Fortran IV
 - 1960-62 arasında gelişti
 - Mantıksal seçim ifadesi: en önemli değişikliği mantıksal *if* ifadesi ve fonksiyonlara başka fonksiyonların parametre olarak aktarılabilmesiydi.
 - Explicit type bildirimleri (örneklerin ilki implicit ve ikincisi explicit)
 - IMPLICIT REAL(A-H)
 - INTEGER :: RESULT
 - Alt program adları parametreler olabilir
 - 1966'da ANSI standardı.
- ◎ 1966'da *Fortran 66* adı altında standart haline geldi(ANSI, 1966).

Fortran 77

- ◎ 1978'de yeni standart oldu
 - Karakter dizisi işleme (string)
 - Mantıksal döngü kontrol ifadesi
 - IF-THEN-ELSE ifadesi
 - Taşınabilirlik daha iyi hale getiriliyor.

Fortran 90

© Fortran 77'den sonra en önemli değişiklikler

- Modüller
- Dinamik diziler
- İşaretçiler (Pointer)
- Özyineleme
- CASE ifadesi
- Parametre tipi kontrolü

Fortran: Diğer versiyonlar

- ◎ Fortran 95 - nispeten küçük eklemeler, artı bazı silmeler
 - Yeni bir iterasyon yöntemi: forall, paralelleştirme görevini kolaylaştırmak için çıkmış bir döngü yöntemi
 - `FORALL (i=1:n,j=1:n,i/=j) A(i,j) = REAL(i+j)`
- ◎ Fortran 2003 - OOP desteği, prosedür işaretçileri, C ile birlikte çalışabilirlik, Veri işleme geliştirmeleri, Giriş / Çıkış geliştirmeleri, Uluslararası kullanım desteği, Bilgisayar işletim sistemiyle gelişmiş entegrasyon
- ◎ Fortran 2008 - yerel kapsamlar (scopes) için bloklar, ortak diziler, Eşzamanlı Yapma (Do Concurrent)
- ◎ Fortran 2018 - C ile Daha Fazla Birlikte Çalışabilirlik, Ek Paralel Özellikler

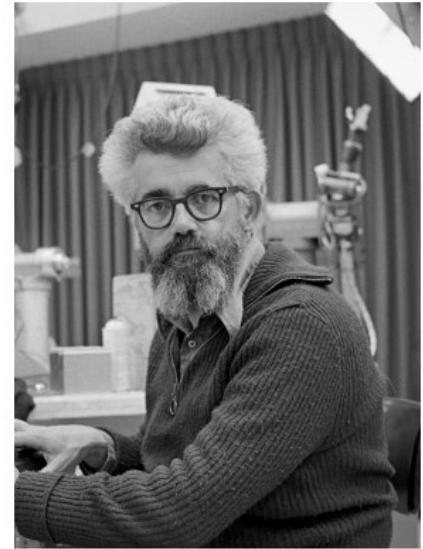
Fortran

- ◎ Birçok programlama diline ışık tutmuştur.
- ◎ Yıllarca bilimsel hesaplamanın ana dili oldu.
- ◎ Astronomi, iklim modelleme, hesaplamalı kimya, hesaplamalı ekonomi, hesaplamalı akışkanlar dinamiği, hesaplamalı fizik, veri analizi, hidrolojik modelleme, sayısal doğrusal cebir ve sayısal kütüphaneler (LAPACK, IMSL ve NAG), optimizasyon, uydu simülasyonu, yapısal mühendislik ve hava tahmini kullanıldı.

Fonksiyonel Programlama: Lisp

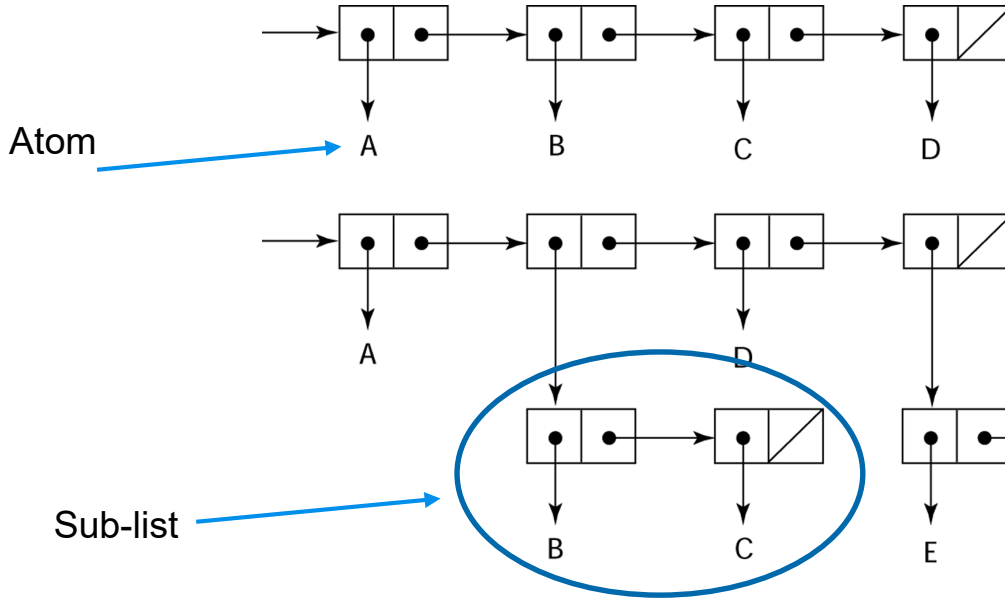
```
(print "Hello world")
```

- ◎ **LISt Processing language**
 - MIT'de 1958 yılında John McCarthy'in geliştirildi.
- ◎ Yapay zeka araştırması için bir dile ihtiyaç vardı.
 - Listelerdeki verileri işleyen (diziler yerine)
 - Sembolik hesaplama (sayısal yerine, yapay zekaya daha uygun, çünkü bir insan sayılarla düşünmez, sembol ve sembol ifadelerini ilişkisi)
- ◎ Yalnızca iki veri türü: atomlar ve listeler
 - Veri tiplmesi bakımından çok esnektir
- ◎ Sözdizimi lambda hesabına (*lambda calculus*) dayanır. Başka bir deyişle parantez kullanımına dayanır.



John McCarthy,
"History of LISP" 1978

Fonksiyonel Programlama: Lisp



Bağlı Liste notasyonu
(A B C D)

A fonksiyon ismi, diğerleri parametreler

(A (B C) D (E (F G)))

Lisp : Basit bir kod

```
; Lisp Example function
; The following code defines a Lisp predicate function
; that takes two lists as arguments and returns True
; if the two lists are equal, and NIL (false) otherwise
(DEFUN equal_lists (lis1 lis2)
  (COND
    ((ATOM lis1) (EQ lis1 lis2))
    ((ATOM lis2) NIL)
    ((equal_lists (CAR lis1) (CAR lis2))
     (equal_lists (CDR lis1) (CDR lis2)))
    (T NIL)
  )
)
```

Fonksiyonel Programlama: Lisp

```
(print "Hello world")
```

- LISP esas olarak yorumlayıcı kullanan bir dildir. Derleyici kullanan versiyonları da vardır.
- Öncü fonksiyonel programlama
 - Değişkenlere veya atamaya gerek yok
 - Özyineleme ve koşullu ifadelerle kontrol
- AI için baskın dil Common Lisp ve Scheme, Lisp'in torunlarıdır. ML, Haskell ve F# aynı zamanda fonksiyonel programlama dilleridir, ancak çok farklı sözdizimi kullanırlar.
- Olağanüstü esneklik, ifade gücü ve **kodun aynı zamanda veri olarak da kullanılabilmesi** özelliği LISP'i yapay zaka uygulamalarında rakipsiz hale getirmiştir.
- Nesne Yönelimli Programlamayı destekler
- Veri tabanlarına erişim ve GUI olanağı vardır.
- Çok iyi belgelendirilmiş olup, COMMON LISP, ANSI tarafından standart hale getirilmiştir.



MIT Müzesindeki
Lisp Makinası

Scheme

```
(let ((hello0 (lambda() (display "Hello world") (newline))))  
  (hello0))
```

- ⊙ Lisp programlama dilleri ailesinin minimalist bir lehçesidir.
- ⊙ 1970'in ortalarında Guy L. Steele ve Gerald Jay Sussman tarafından MIT'de geliştirilmiştir.
- ⊙ Küçük
- ⊙ Scheme dilinde prefix ifadeler kullanılır.
 - $(+ 3 5) = 3 + 5 = 8$
 - $(+ 3 (* 4 5)) = 3 + 4 * 5 = 23$
- ⊙ Basit sözdizimi (ve küçük boyutu), eğitim uygulamaları için idealdir

Common Lisp

- ◎ Lisp'in çeşitli lehçelerinin özelliklerini tek bir dilde birleştirme çabası
- ◎ Endüstride bazı büyük uygulamalar için kullanılır
- ◎ Common Lisp, genel amaçlı, çok paradigmatlı bir programlama dilidir.
 - Prosedürel, fonksiyonel ve nesneye yönelik programlama paradigmalarının bir kombinasyonunu destekler.
 - Veri türleri: Records, arrays, complex numbers, and character strings
- ◎ Dinamik bir programlama dili olarak, verimli çalışma zamanı programlarına yinelemeli derleme ile evrimsel ve artımlı yazılım geliştirmeyi kolaylaştırır.
- ◎ Örnek Kod: Sonuç = 10, mantığını çözmeye çalışın

```
(let ((a 6)
      (b 4))
  (+ a b))
```

https://en.wikipedia.org/wiki/Common_Lisp

Fonksiyonel Programlama Dilleri: Diğerleri: ML

- © ML (MetaLanguage; Ullman, 1998) ilk olarak 1980'lerde Edinburgh Üniversitesi'nde Robin Milner tarafından Logic for Computable Functions adlı bir program doğrulama sistemi için bir metal dil olarak tasarlandı.
- © Emir esaslı programlamayı da destekler.
- © Çalışma zamanında karar verir. (compile time)
- © Sözdizimi diğer yaklaşımlara benzemez.

Fonksiyonel Programlama Dilleri:

Diğerleri: Miranda, Caml, OCaml, F#, Clojure

- ◎ Miranda, 1980'lerin başında İngiltere - Kent Üniversitesi'nde David Turner (1986) tarafından geliştirildi.
 - Günümüzün popüler fonksiyonel programlama dili Haskell Miranda tabanlıdır.
 - Purely Fonksiyonel Programlama: atama ve değişkenler yok
- ◎ Caml (Cousineau ve diğerleri, 1998) ve nesne yönelimli programlamayı destekleyen lehçesi geliştirmiştir.
- ◎ OCaml (Smith, 2006), ML ve Haskell'den gelmektedir.
- ◎ F #, doğrudan OCaml'ye dayalı nispeten yeni bir yazım dilidir. F#: .NET çerçevesinde çalışan, fonksiyon odaklı bir programlama dilidir. (2005)
- ◎ Clojure: Lisp'in modern bir yorumu olması amaçlanan fonksiyonel bir programlama dilidir. (2007)

Fonksiyonel Programlamada Yeni Trend: Haskell

```
module Main (main) where

main :: IO ()
main = putStrLn "Hello, World!"
```

- Haskell, genel amaçlı ve tamamen fonksiyonel bir programlama dilidir.
- Haskell'in ana uygulaması Glasgow Haskell Derleyicisidir (GHC). Mantıkçı Haskell Curry'nin adını almıştır.
- İlk versiyonu, Philip Wadler ve Stephen Blott tarafından 1987 geliştirilmiştir.
- Öğretim, araştırma ve endüstriyel uygulama için uygun olacak şekilde geliştirilen Haskell, tip sınıfları gibi tip güvenli operatör aşırı yüklemesini mümkün kılan bir dizi gelişmiş programlama dili özelliğine öncülük etmiştir.
- Haskell, lazy evulation kullanır. Lazy evulation: değeri gerekmedikçe hiçbir ifadenin değerlendirilmediği anlamına gelir.

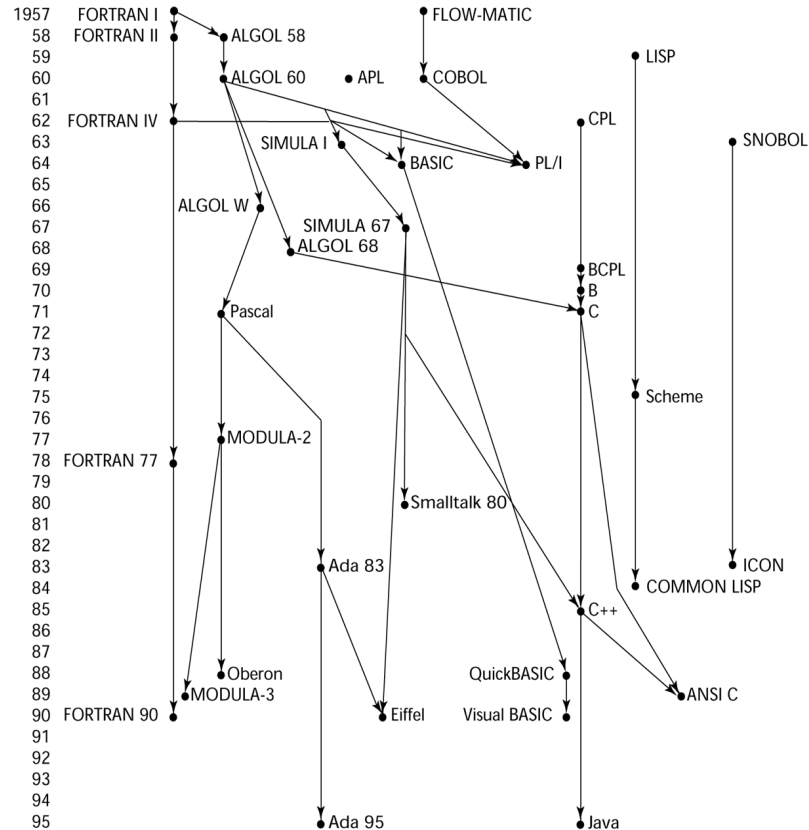


Gelişmişliğe Doğru İlk Adım: ALGOL 60

```
BEGIN DISPLAY("HELLO WORLD!") END.
```

- ◎ **ALGO**rithmic **L**anguage
- ◎ Sonraki dilleri en çok etileyen programlama dilidir.
- ◎ Geliştirme ortamı FORTRAN, IBM 70x için (zar zor) gelmişti
 - Hepsi belirli makineler için birçok başka dil geliştiriliyordu
 - Taşınabilir dil yok; hepsi makineye bağımlıydı
 - İletişim algoritmaları için evrensel bir dil yok
- ◎ ALGOL 60, evrensel bir dil tasarlama çabalarının sonucuydu
- ◎ FORTRAN I'den esinlenilmiştir
- ◎ İlk kez 1958 yılında Avrupalı ve Amerikalı bir komisyonun Zürih'teki çalışmaları sonucu oluşturulan yüksek seviyeli ve emir esaslı bir dildir. (İlk ismi ALGOL 58)

Gelişmişliğe Doğru İlk Adım: ALGOL 60



İlk tasarım süreçleri

- ◎ ACM ve GAMM, tasarım için dört gün bir araya geldi (27 Mayıs - 1 Haziran 1958, Zürih)
- ◎ Dilin hedefleri
 - Matematiksel gösterime yakın ve kolay okunabilen
 - Algoritmaları açıklamak için iyidir
 - Makine koduna çevrilebilir olmalıdır
- ◎ Makineden bağımsız, daha esnek ve daha güçlü bir dil tasarımı olmuştur.

Association for Computing Machinery (ACM)

German Gesellschaft für Angewandte Mathematik und Mechanik (GAMM)

ALGOL 58

- ⊙ Type kavramı formalize edildi.
- ⊙ İsimler herhangi bir uzunlukta olabilir
- ⊙ Diziler herhangi bir sayıda aboneye sahip olabilir
- ⊙ Parametreler moda göre ayrıldı (giriş ve çıkış)
- ⊙ Alt karakter parantez içine yerleştirildi
- ⊙ Bileşik ifadeler (begin ... end)
- ⊙ İfade ayırıcı olarak noktalı virgül
- ⊙ Atama operatörü şuydu :=
- ⊙ eğer else-if cümlesi geldi
- ⊙ G / Ç yok - "makineye bağımlı hale getirir"

ALGOL 58 Uygulaması

- ◎ Uygulanması geliştirilmemiştir, ancak varyasyonları (MAD, JOVIAL)
- ◎ IBM başlangıçta hevesli olmasına rağmen, tüm destek 1959 ortalarında kesildi

ALGOL 60

- ◎ ALGOL 58, Paris'teki 6 günlük toplantıda değiştirildi
- ◎ Yeni özellikler
 - Block structure (local scope)
 - İki parametrelili taşıma metodu (in, out)
 - Alt program özyinelemesi
 - Stack-Dinamik Diziler
 - Hala G / Ç yok ve string işleme yok

ALGOL 60 Başarıları

- ◎ 20 yılı aşkın süredir algoritmaları yayınlamanın standart yoluydu
- ◎ Sonraki tüm emir esaslı diller buna dayanmaktadır
- ◎ İlk makineden bağımsız dil, sözdizimi resmi olarak tanımlanan birinci dil (BNF - Backus Naur Form, sözdizimi tanımlama standardı)

ALGOL 60 Başarısızlıkları

- ◎ Asla yaygın olarak kullanılmadı, (özellikle ABD'de)
- ◎ Sebepler
 - G / Ç eksikliği ve karakter seti programları taşınabilir değil
 - Çok esnek - uygulaması zor
 - Formal sözdizimi açıklaması: BNF kullanımı karmaşıklıklaştırmış gibi gözükmemektedir.
 - IBM'den destek eksikliği
 - Çünkü bu sırada IBM FORTRAN dilinde yazılmış zengin bir kütüphaneye sahiptir ve haliyle FORTRAN dilini desteklemektedir.

Bir Algol Kod Örneği

```
comment ALGOL 60 Example Program
Input:  An integer, listlen, where listlen is less than
        100, followed by listlen-integer values
Output: The number of input values that are greater than
        the average of all the input values ;

begin
  integer array intlist [1:99];
  integer listlen, counter, sum, average, result;
  sum := 0;
  result := 0;
  readint (listlen);
  if (listlen > 0) ^ (listlen < 100) then
    begin
comment Read input into an array and compute the average;
      for counter := 1 step 1 until listlen do
        begin
          readint (intlist[counter]);
          sum := sum + intlist[counter]
        end;
comment Compute the average;
      average := sum / listlen;
comment Count the input values that are > average;
      for counter := 1 step 1 until listlen do
        if intlist[counter] > average
          then result := result + 1;
comment Print result;
      printstring("The number of values > average is:");
      printint (result)
    end
  else
    printstring ("Error-input list length is not legal");
  end
end
```

İşletme Kayıtlarını Bilgisayarlaştırma: COBOL

- ◎ **CO**mmun **B**usiness **O**riented **L**anguage
- ◎ İngilizce sözcük yada cümle yapılarını kullanan ve **İşletmelere** yönelik ortak bir dildir.
- ◎ Büyük miktarda bilgi G-Ç yapıldığı uygulamalar için geliştirilmiştir.
- ◎ Benzer diller
 - UNIVAC, FLOW-MATIC kullanmaya başlıyordu
 - USAF, AIMACO kullanmaya başlıyordu
 - IBM, COMTRAN'ı geliştiriyordu

COBOL

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. hello-world.  
PROCEDURE DIVISION.  
    DISPLAY "Hello, world!"  
.
```

- ◎ FLOW-MATIC (yaklaşık 20 komuttan oluşan bir dil) üzerine kurulmuştur
- ◎ FLOW-MATIC özellikleri
 - 12 karaktere kadar adlar
 - Aritmetik operatörler için İngilizce isimler (aritmetik ifadeler yok)
 - Veriler ve kod tamamen ayrıydı
 - Her ifadedeki ilk kelime bir fiildi
- ◎ Fonksiyonları desteklememekte idi

COBOL Tasarım Süreci

- ◎ İlk Tasarım Toplantısı (Pentagon) - Mayıs 1959
- ◎ Tasarım hedefleri
 - Basit bir İngilizce gibi görünmeli
 - Daha az güçlü olacağı anlamına gelse bile, kullanımı kolay olmalı
 - Bilgisayar kullanıcılarının tabanını genişletmeli
 - Mevcut derleyici sorunları tarafından önyargılı olmamalıdır
- ◎ Tasarım komitesi üyelerinin tamamı bilgisayar üreticilerinden ve Savunma Bakanlığı şubelerinden
- ◎ Tasarım Problemleri: aritmetik ifadeler? Alt kelimeler? Üreticiler arasında kavgalar

COBOL Değerlendirmesi

- ⊙ Üst düzey bir dilde ilk makro kullanan dil (DEFINE)
- ⊙ Hiyerarşik veri yapıları (kayıtlar - records)
- ⊙ İç içe geçmiş seçim ifadeleri
- ⊙ Uzun adlar (30 karaktere kadar)
- ⊙ Ayrı veri bölümü: değişken bu bölüme tanımlanır. Prosedür bölümü ise zayıftır.
- ⊙ Department of Defense (**DOD**) etkisinde kalmıştır.
- ⊙ COBOL61-65-70-73, ANSI-COBOL versiyonları bulunmaktadır
- ⊙ 1990'larda ise OOP versiyonu üretilmiştir.
- ⊙ Üzerinde hala çalışmalar yapılmakta ve Amerika hala büyük miktardaki verileri işlerken COBOL kullanmaktadır.

Cobol Kod Örneği

```
1 IDENTIFICATION DIVISION.  
2 PROGRAM-ID. VARS.  
3  
4 DATA DIVISION.  
5     *> working storage defines variables  
6     WORKING-STORAGE SECTION.  
7     *> define a number with a sign, 3 numbers, a decimal, and then  
8     *> two numbers aafter the decimal. by default it should be 0 filled  
9     01 FIRST-VAR PIC S9(3)V9(2).  
10    *> do the same thing as above but actually initialize  
11    *> to a number -123.45  
12    01 SECOND-VAR PIC S9(3)V9(2) VALUE -123.45.  
13    *> defines an alphabetic string and initialize it to abcdef  
14    01 THIRD-VAR PIC A(6) VALUE 'ABCDEF'.  
15    *> define an alphanumeric string and initialize it to a121$  
16    01 FOURTH-VAR PIC X(5) VALUE 'A121$'.  
17    *> create a grouped variable  
18    01 GROUP-VAR.  
19        05 SUBVAR-1 PIC 9(3) VALUE 337.  
20    *> create 3 alphanumerics, but use less than  
21    *> the allocated space for each of them  
22    05 SUBVAR-2 PIC X(15) VALUE 'LALALALA'.  
23    05 SUBVAR-3 PIC X(15) VALUE 'LALALA'.  
24    05 SUBVAR-4 PIC X(15) VALUE 'LALALA'.  
25  
26    *> print our variables  
27    PROCEDURE DIVISION.  
28    DISPLAY "1ST VAR : "FIRST-VAR.  
29    DISPLAY "2ND VAR : "SECOND-VAR.  
30    DISPLAY "3RD VAR : "THIRD-VAR.  
31    DISPLAY "4TH VAR : "FOURTH-VAR.  
32    DISPLAY "GROUP VAR : "GROUP-VAR.  
33    STOP RUN.
```



```
10 PRINT "Hello, World!"  
20 END
```

Zaman Paylaşımının Başlangıcı: Basic

- ◎ **Beginner's All-purpose Symbolic Instruction Code**
- ◎ 1964'te Dartmouth'ta Kemeny & Kurtz tarafından tasarlandı
- ◎ Tasarım Hedefleri:
 - Fen bilgisi olmayan öğrenciler için öğrenmesi ve kullanması kolay
 - "Hoş ve arkadaş canlısı" olmalı
 - Ev ödevleri için hızlı geri dönüş
 - Ücretsiz ve özel erişim
 - Kullanıcı zamanı bilgisayar saatinden daha önemlidir
 - Zaman paylaşımı ile yaygın olarak kullanılan ilk dil
- ◎ Sadece 14 komuta (LET, PRINT, GOTO...) sahipti. Tek veri tipi (number= kayan noktalı ve tamsayı)
- ◎ FORTRAN'dan DO çevrimini, ALGOL'den ise "until" yerine "to" gibi deyimleri almıştır.

Basic (1964)

- ◎ **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode
- ◎ 1964'te John Kemeny ve Thomas Kurtz tarafından yapılmıştır.
- ◎ Kolay bir dil ve genel maksatlı, belirli bir alana bağlı değil
- ◎ Uzman kişilere de hitap edebiliyor, öğrencilerin öğrenmesi ve ödevleri için ideal
- ◎ Açık ve anlaşılır hata mesajlarına sahip, kullanıcı bilgisayarla etkileşimli çalışabiliyor
- ◎ Küçük boyutlu programları hızlı bir biçimde çalıştırabiliyor
- ◎ Kullanım için donanım bilgisine sahip olmaya gerek yok
- ◎ Kullanıcıyı işletim sistemi ayrıntılarından dahi koruyabiliyor
- ◎ Derleyici kullanıyor, programın tümü makine diline çevrildikten sonra icra ediliyor
- ◎ BASIC'in pek çok versiyonları olmuştur. 1989'da ise nesne yönelimli uyarlama olan Visual BASIC ve 1998'de VB6.0 sunulmuştur.



Basic Kod Örneği

```
REM Basic Example Program
REM Input: An integer, listlen, where listlen is less
REM         than 100, followed by listlen-integer values
REM Output: The number of input values that are greater
REM         than the average of all input values
DIM intlist(99)
result = 0
sum = 0
INPUT listlen
IF listlen > 0 AND listlen < 100 THEN
REM Read input into an array and compute the sum
  FOR counter = 1 TO listlen
    INPUT intlist(counter)
    sum = sum + intlist(counter)
  NEXT counter
REM Compute the average
  average = sum / listlen
REM Count the number of input values that are > average
  FOR counter = 1 TO listlen
    IF intlist(counter) > average
      THEN result = result + 1
  NEXT counter
REM Print the result
  PRINT "The number of values that are > average is:";
    result
ELSE
  PRINT "Error-input list length is not legal"
END IF
END
```

Herkes İçin Her Şey: PL / I (1965)

```
HELLO:  PROCEDURE OPTIONS (MAIN);

        /* A PROGRAM TO OUTPUT HELLO WORLD */
        FLAG = 0;

LOOP:    DO WHILE (FLAG = 0);
          PUT SKIP DATA('HELLO WORLD!');
        END LOOP;

END HELLO;
```

- ◎ IBM ve SHARE tarafından tasarlandı
- ◎ COBOL, Fortran IV ve ALGOL 60'tan sonra bilimsel ve işletme problemlerine çözüm sağlayabilmek için floating-point ve decimal veri tiplerini desteklemek üzere geliştirilmiş bir dildir.
- ◎ 1964'teki bilgi işlem durumu (IBM'in bakış açısı)
 - Bilimsel hesaplama
 - ◎ IBM 1620 ve 7090 bilgisayarlar
 - ◎ FORTRAN
 - ◎ SHARE kullanıcı grubu
 - İşletme bilgi işlem
 - ◎ IBM 1401, 7080 bilgisayarlar
 - ◎ COBOL
 - ◎ GUIDE kullanıcı grubu

Herkes İçin Her Şey: PL / I

- ◎ Önerdiği çözüm: (Bilimsel ve Mesleki uygulamalar)
 - Her iki tür uygulamayı da yapmak için yeni bir bilgisayar oluşturun
 - Her iki tür uygulamayı da yapmak için yeni bir dil tasarlayın
- ◎ Farklı programlama dillerinin güzel özelliklerini bir araya getirmeyi amaçlamıştır.
 - ALGOL 60 (recursion and block structure),
 - Fortran IV (global data aracılığıyla iletişim ile ayrı derleme),
 - COBOL 60 (data structures, input/output, and reportgenerating facilities)
- ◎ Teoride iyi olsa da pek kullanımı olmamıştır.

Herkes İçin Her Şey: PL / I

- ◎ Gelişimi olmasa da literatüre katkıları vardır:
 - Programların eşzamanlı olarak yürütülen alt programlar oluşturmaya izin verildi. Bu iyi bir fikir olmasına rağmen, PL / I'de bu özellik zayıf kaldı.
 - 23 farklı türdeki **istisnayı** veya çalışma zamanı hatalarını tespit etmek ve işlemek mümkündü.
 - Alt programların yinelemeli olarak kullanılmasına izin verildi. Diğer taraftan daha verimli bir sistem için bu özelliğin devre dışı bırakılmasına da olanak verildi.
 - İşaretçiler bir veri türü olarak dahil edildi.
 - Dizilerin kesitlerine referans verilebilir. Örneğin, bir matrisin üçüncü satırına tek boyutlu bir diziymiş gibi başvurulabilir.

◎ Problemler

- Karmaşıklık
- Kötü tasarım

PL / I Kod Örneği

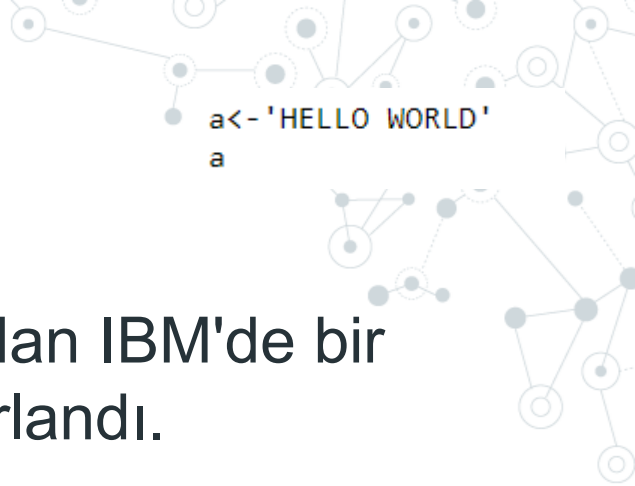
```
/* PL/I PROGRAM EXAMPLE
INPUT:  AN INTEGER, LISTLEN, WHERE LISTLEN IS LESS THAN

        100, FOLLOWED BY LISTLEN-INTEGER VALUES
OUTPUT: THE NUMBER OF INPUT VALUES THAT ARE GREATER THAN
        THE AVERAGE OF ALL INPUT VALUES */
PLIEX: PROCEDURE OPTIONS (MAIN);
  DECLARE INTLIST (1:99) FIXED.
  DECLARE (LISTLEN, COUNTER, SUM, AVERAGE, RESULT) FIXED;
  SUM = 0;
  RESULT = 0;
  GET LIST (LISTLEN);
  IF (LISTLEN > 0) & (LISTLEN < 100) THEN
    DO;
/* READ INPUT DATA INTO AN ARRAY AND COMPUTE THE SUM */
      DO COUNTER = 1 TO LISTLEN;
        GET LIST (INTLIST (COUNTER));
        SUM = SUM + INTLIST (COUNTER);
      END;
/* COMPUTE THE AVERAGE */
      AVERAGE = SUM / LISTLEN;
/* COUNT THE NUMBER OF VALUES THAT ARE > AVERAGE */
      DO COUNTER = 1 TO LISTLEN;
        IF INTLIST (COUNTER) > AVERAGE THEN
          RESULT = RESULT + 1;
      END;
/* PRINT RESULT */
      PUT SKIP LIST ('THE NUMBER OF VALUES > AVERAGE IS:');
      PUT LIST (RESULT);
    END;
  ELSE
    PUT SKIP LIST ('ERROR-INPUT LIST LENGTH IS ILLEGAL');
  END PLIEX;
```


İki Erken Dinamik Dil (1960'lar): APL ve SNOBOL

- ⦿ Dinamik yazım ve dinamik depolama tahsisi ile karakterize edilmiştir.
- ⦿ Değişkenler türsüzdür
 - Bir değişken, kendisine bir değer atandığında bir tür edinir
- ⦿ Depolama, bir değer atandığında bir değişkene tahsis edilir. (Bellek daha dinamik kullanılıyor)

APL



```
a ← 'HELLO WORLD'
a
```

- ◎ 1960 civarında Ken Iverson tarafından IBM'de bir donanım tanımlama dili olarak tasarlandı.
- ◎ Dil ifadelerini oluşturan pek çok sayıda operatör vardır. Ancak, okuması oldukça zordur. Diğer taraftan kodların yazılması hızlıdır.
- ◎ Çok az da olsa hala kullanımda.

SNOBOL

```
* SNOBOL program to print Hello World
  I = 1
LOOP  OUTPUT = "Hello, world!"
      I = I + 1
      LE(I, 5) : S(LOOP)

END
```

- ◎ 1964 yılında Farber, Griswold, and Polensky tarafından SNOBOL (Snowball) metin işleyici (text processing) olarak Bell Lab'da özel olarak üretilmiştir.
- ◎ String pattern eşleştirmede güçlü operatörler
- ◎ Alternatiflerine göre yavaş bir dildir.
- ◎ SNOBOL dili yorumlayıcı kullanan bir dildir.
- ◎ Veri ve değişkenler üzerinde tip bakımından bir sınırlama bulunmuyordu.
- ◎ Metin işlemede hala kullanılır.
- ◎ Bu dilin devamı: awk, icon, perl,

Veri Soyutlamanın Başlangıcı: SIMULA 67

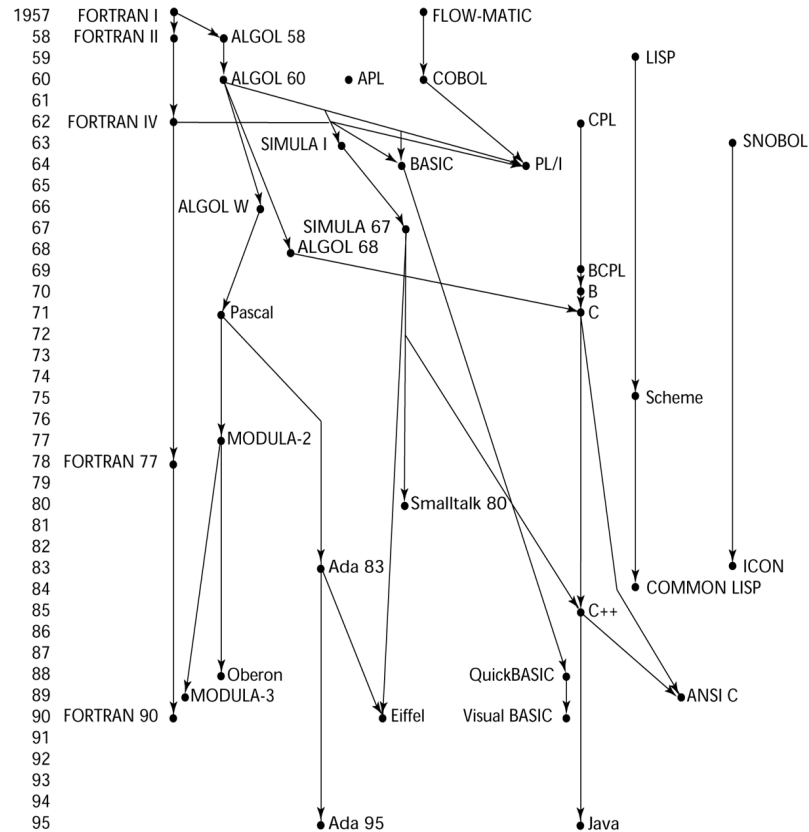
- 1964 yılında Norveç'te Kristen Nygaard ve Ole-Johan Dahl ilk olarak simülasyon için tasarlanmış ve uygulaması çıkmış bir dildir.
- ALGOL60'ın tabanını kullanır. (Blok yapısı ve kontrol ifadeleri buradan alınmıştır)
- Yaygın bir kullanıma ulaşmasa da iki ana katkısı ön plana çıkar:
 - Coroutines - bir tür alt program: Daha önce durdurulduğu yerden itibaren yeniden çalışmaya başlayan altprogramları desteklemektedir.
 - Algol 60'ın etkin olmadığı konulardan biri alt programların tasarımı idi.
 - Simülasyon, daha önce durdukları konumda yeniden başlamalarına izin verilen alt programlar gerektirir.
 - Sınıflar, nesneler ve miras : Veri soyutlamasına imkan veren sınıf yapılarını desteklemektedir. (Bu konudaki ilk adımlardan) Veri soyutlaması, OOP temel taşlarından biridir. Hoare (1972) kadar çalışılmış bir konu değildir.

```
! Hello World in Simula;  
  
BEGIN  
  OutText("Hello World!");  
  OutImage;  
END
```



Kristen Nygaard and Ole-Johan Dahl
"The Development of SIMULA
Languages", 1978

Tarihçe



Ortogonal Tasarım: ALGOL 68

```
printf(($gl$, "Hello, World!"))
```

- ◎ ALGOL 60'ın sürekli geliştirilmesinden ancak bu dilin bir tam özelliklerini destekleyen bir dil değildir.
- ◎ Birkaç yeni fikrin kaynağı (dilnin kendisi hiçbir zaman yaygın kullanıma ulaşmamış olsa da)
 - Kullanıcı tanımlı(user-defined) veri tiplerini destekler
 - ALGOL68'de flex adı verilen dinamik dizi kavramına izin verir
 - Tasarım, ortogonalite kavramına dayanmaktadır
- ◎ Zorlukları
 - Öğrenilmesi oldukça zor olan bir gramer (BNF'ye dayanmıyor, van Wijngaarden gramer yapısı var) ve dil yapısına sahiptir. Bu açıdan ALGOL 60'a göre az kullanılmıştır.
 - Sadece bilimsel uygulamalar hedeflenerek tasarlanmış bir dil
- ◎ Pascal, C ve Ada gibi dilleri etkilemiştir.
- ◎ PL / I aynı dönemde IBM desteği ile daha fazla kullanılmıştır.

ALGOL'un Torunları:

Pascal (1971)

- ◎ Niklaus Wirth tarafından geliştirildi (ALGOL 68 komitesinin eski bir üyesi)
- ◎ Pascal adını, 1642'de ilk mekanik toplama makinesini icat eden Fransız filozof ve matematikçisi olan Blaise Pascal'dan almıştır.
- ◎ Yapısal programlamayı öğretmek için tasarlandı
- ◎ Basit ve okunabilir, yazılabilir
- ◎ Fortran and C ile karşılaştırıldığında güvenli (safety) bir dildir.
- ◎ En büyük etki, programlama öğretiminde oldu
 - 1970'lerin ortalarından 1990'ların sonuna kadar, programlama öğretmek için en yaygın kullanılan dildi

```
program HelloWorld;  
uses crt;  
  
(* Here the main program block starts *)  
begin  
    writeln('Hello, World!');  
    readkey;  
end.
```



ALGOL'ün Torunları:

Pascal (1971)

```
{Pascal Example Program
Input:  An integer, listlen, where listlen is less than
        100, followed by listlen-integer values
Output: The number of input values that are greater than
        the average of all input values }
program pasex (input, output);
  type intlisttype = array [1..99] of integer;
  var
    intlist : intlisttype;
    listlen, counter, sum, average, result : integer;
  begin
    result := 0;
    sum := 0;
    readln (listlen);
    if ((listlen > 0) and (listlen < 100)) then
      begin
        { Read input into an array and compute the sum }
        for counter := 1 to listlen do
          begin
            readln (intlist[counter]);
            sum := sum + intlist[counter]
          end;
        { Compute the average }
        average := sum / listlen;
        { Count the number of input values that are > average }
        for counter := 1 to listlen do
          if (intlist[counter] > average) then
            result := result + 1;
        { Print the result }
        writeln ('The number of values > average is:',
                  result)
      end { of the then clause of if (( listlen > 0 ... )
    else
      writeln ('Error-input list length is not legal')
    end.
```


ALGOL'ün Torunları:

Bir taşınabilir sistem dili :C (1972)

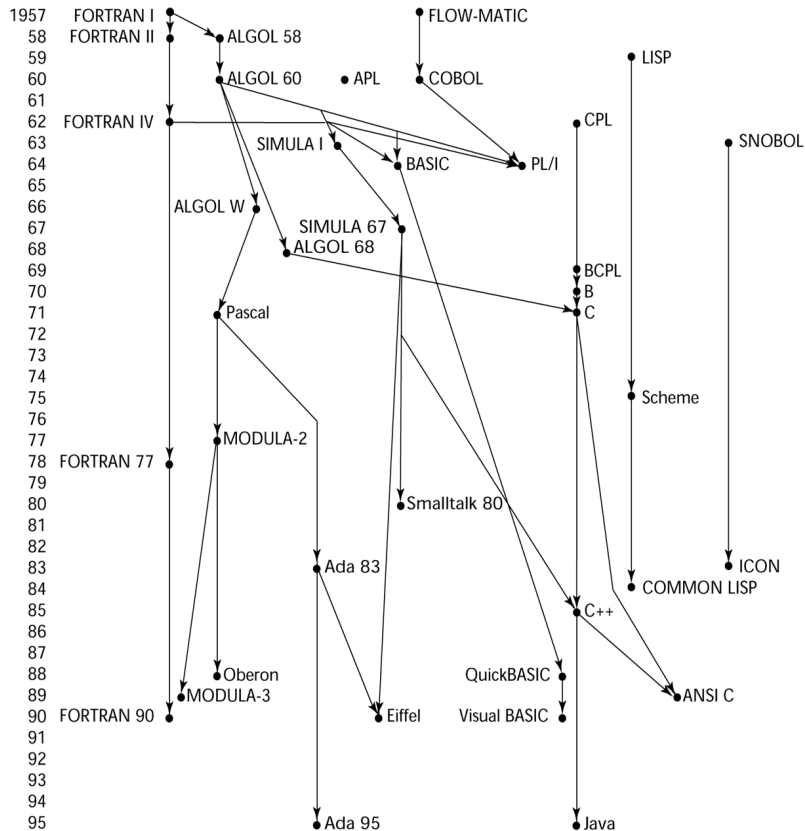
- Sistem programlama için tasarlanmıştır (Bell Labs'de Dennis Ritchie ve Ken Thompson)
 - Öncelikle BCLP ve B'den, aynı zamanda ALGOL 68'den alıntılar yaptı.
 - BCLP ve B'den örneğin kayan noktalı veri tipi
 - ALGOL 68'den for ve switch, atama operatörleri, pointer
- Güçlü operatör seti
- Tip kontrolü yetersiz
- Başlangıçta UNIX aracılığıyla yayıldı
- Sistem dili olarak tasarlanmış olmasına rağmen birçok uygulama alanında yaygın olarak kullanılmıştır.
- 1978 kadar ufak geliştirmeler devam etti, bu yıl ortasında çıkan kitapta ilk defa geniş şekilde tanıtıldı.
- 1989'da ANSI, uygulayıcıların dile dahil etmiş olduğu birçok özelliği içeren resmi bir C tanımını (ANSI, 1989) üretti. (C89)
- 1999 ISO ile tanım güncellendi. (C99)

```
#include <stdio.h>
int main() {
    // printf() displays the string inside quotation
    printf("Hello, World!");
    return 0;
}
```



Dennis Ritchie (Sağdaki)
Ken Thompson

Tarihçe



**Ken Thompson,
Dennis Ritchie**
"The Development of
the C Language" 1993

C Kod Örneği

```
/* C Example Program
Input:  An integer, listlen, where listlen is less than
        100, followed by listlen-integer values
Output: The number of input values that are greater than
        the average of all input values */
int main (){
    int intlist[99], listlen, counter, sum, average, result;
    sum = 0;
    result = 0;
    scanf("%d", &listlen);
    if ((listlen > 0) && (listlen < 100)) {
        /* Read input into an array and compute the sum */
        for (counter = 0; counter < listlen; counter++) {
            scanf("%d", &intlist[counter]);
            sum += intlist[counter];
        }
        /* Compute the average */
        average = sum / listlen;
        /* Count the input values that are > average */
        for (counter = 0; counter < listlen; counter++)
            if (intlist[counter] > average) result++;
        /* Print result */
        printf("Number of values > average is:%d\n", result);
    }
    else
        printf("Error-input list length is not legal\n");
}
```

Mantığa Dayalı Programlama: Prolog (1972)

```
parent(person("Bill", "male"), person("John", "male")).  
parent(person("Pam", "female"), person("Bill", "male")).  
parent(person("Pam", "female"), person("Jane", "female")).  
parent(person("Jane", "female"), person("Joe", "male")).
```

```
grandFather(Person, TheGrandFather) :-  
    parent(Person, ParentOfPerson),  
    father(ParentOfPerson, TheGrandFather).
```

```
father(P, person(Name, "male")) :-  
    parent(P, person(Name, "male")).
```

- ◎ **PRO**gramming **LOG**ic
- ◎ Comerauer ve Roussel (Aix-Marseille Üniversitesi) tarafından, Kowalski'nin (Edinburgh Üniversitesi) yardımıyla geliştirildi
- ◎ Biçimsel mantığa (formal logic) dayalı
- ◎ Prosedürel bir yaklaşım değil
- ◎ Bir nesneler kümesi ile bu nesnelerle ilişkili hedeflere nasıl erişilebileceğini tanımlayan kurallar kümesinden oluşur.
- ◎ Verilen sorguların doğruluğunu anlamak için bir çıkarım sürecini kullanan akıllı bir veritabanı sistemi olarak özetlenebilir.
- ◎ Nispeten verimsiz
- ◎ Birkaç uygulama alanı: VTYS ve Yapay Zeka ile sınırlı
- ◎ 1994 yılında OOP desteği olan Prolog++ tanıtılmıştır.

Mantığa Dayalı Programlama: Prolog (1972)

- ◎ Prolog fact ve rule'lardan oluşur.
- ◎ Fact tarafı:
 - anne(Ayşe, Ali).
 - baba(Metin, Ayşe).
- ◎ Rule tarafı:
 - Büyük_ebeveyn(X, Z) :- ebeveyn(X, Y), ebeveyn(Y, Z).
- ◎ Goal tarafı:
 - baba(Ali, Feyyaz)
 - Resolution process yapılır, önceki fact ve rule'lar incelenip goal sonucu olarak true veya false yapılır.

Tarihin En Büyük Tasarım Çabası: Ada (1983)

- ◎ ABD savunma bakanlığının bir çalışması sonucu ortaya çıkmıştır. Gömülü sistemlerin (embedded systems) programlanmasını sağlayacak PL üretimi amaçlanmıştır. (Department of Defense - DoD)
 - Gömülü sistem, bilgisayar donanımının kontrol ettiği veya hizmet sağladığı cihaza gömülü olduğu bir sistemdir.
 - Önceleri, assembly language yaygın olarak kullanılıyor.
- ◎ Yüzlerce insanı, çok parayı ve yaklaşık sekiz yılı kapsayan büyük tasarım çabasıdır.
- ◎ Gerçek zamanlı uygulamalar için amaçlanmıştır.
- ◎ Matematikçi Augusta **Ada** Byron'ın (1815-1851) ismini almıştır. Analitik makine üzerine çalışmalar yapmış ve bilgisayar tarafından işlenebilecek bir algoritma yazdığı için dünyanın ilk programcısı olarak değerlendirilir.
- ◎ Blok yapılı, nesne yönelimli, genel amaçlı ve eşzamanlılığı destekleyen bir dildir.
- ◎ Büyük boyutlu yazılımlar için uygundur.
- ◎ Gereksinimler dizisi (1975-1978)
 - (Strawman, Woodman, Tinman, Ironman, Steelman)



Ada

```
with Ada.Text_IO;  
  
procedure Hello is  
begin  
    Ada.Text_IO.Put_Line("Hello, world!");  
end Hello;
```

⊙ Katkılar

- Paketler - veri soyutlama desteği
- İstisna işleme - ayrıntılı
- Ada'da program birimleri generic olabilir. Örneğin, sıralanacak veriler için belirtilmemiş bir tür kullanan bir sıralama prosedürü yazmak mümkündür. Tekrar kullanılabilirliği arttıran bir özelliktir.
- Eşzamanlılık - görevlendirme modeli aracılığıyla

⊙ Ek Bilgiler

- Rekabetçi tasarım
- Veri tipleri konusunda çok zengindir.
- Yazılım mühendisliği ve dil tasarımı hakkında ileride kullanılabilecek her şeyi içeriyordu.
- Çok geniş ve çok karmaşık bir dil. (Özellikle yazım kuralları)
- İlk derleyiciler çok zordu; gerçekten kullanılabilir ilk derleyici, dil tasarımı tamamlandıktan yaklaşık beş yıl sonra geldi
- En çok gömülü sistemlerde başarılı olmuştur.

Ada

- ◎ Ada 95: 1988'de geliştirilmeye başlandı
 - Type türetme yoluyla OOP desteği, polymorphism, inheritance, dinamik bağlama
 - Paylaşılan veriler (Shared Data) için daha iyi kontrol mekanizmaları
 - Yeni eşzamanlılık (concurrency) özellikleri
 - Daha esnek kütüphaneler
- ◎ Aynı zamanda ortaya çıkan C++ 'nın popülerliğinden dolayı popüler olamadı.
- ◎ Ada 2005
 - Arayüzler (Interfaces) ve Senkronizasyon (synchronizing) arayüzleri

Ada kod örneği

```
-- Ada Example Program
-- Input:  An integer, List_Len, where List_Len is less
--         than 100, followed by List_Len-integer values
-- Output: The number of input values that are greater
--         than the average of all input values
with Ada.Text_IO, Ada.Integer.Text_IO;
use Ada.Text_IO, Ada.Integer.Text_IO;
procedure Ada_Ex is
  type Int_List_Type is array (1..99) of Integer;
  Int_List : Int_List_Type;
  List_Len, Sum, Average, Result : Integer;
begin
  Result:= 0;
  Sum := 0;
  Get (List_Len);
  if (List_Len > 0) and (List_Len < 100) then
    -- Read input data into an array and compute the sum
    for Counter := 1 .. List_Len loop
      Get (Int_List(Counter));
      Sum := Sum + Int_List(Counter);
    end loop;
    -- Compute the average
    Average := Sum / List_Len;
    -- Count the number of values that are > average
    for Counter := 1 .. List_Len loop
      if Int_List(Counter) > Average then
        Result:= Result+ 1;
      end if;
    end loop;
    -- Print result
    Put ("The number of values > average is:");
    Put (Result);
    New_Line;
  else
    Put_Line ("Error-input list length is not legal");
  end if;
end Ada_Ex;
```

Nesneye Yönelik Programlama : Smalltalk

Transcript show: 'Hello World!'.

- ⊙ Xerox PARC'ta önce Alan Kay tarafından, daha sonra Adele Goldberg tarafından geliştirildi
- ⊙ OOP bir dilin ilk tam uygulaması - 3 temel karakteristiği: veri soyutlama, kalıtım ve dinamik bağlama (data abstraction, inheritance ve dynamic binding)
- ⊙ Smalltalk sadece bir PL değil, aynı zamanda yazılım geliştirme aracıdır. Grafik kullanıcı arayüzü tasarımına öncülük etti.
- ⊙ Program birimleri, verileri kapsülleyen nesneler ve yöntemlerden oluşur.
- ⊙ Hesaplama bir nesneye bir mesaj göndererek yapılır. (alt program)
- ⊙ Kaynak kodunun açık olması iyi bir eğitim ortamı haline gelmesini sağlamıştır. Diğer dillere göre çok basit bir sözdizimi vardır.
- ⊙ Platform bağımsız bir dildir.

Emir Esaslı ve Nesnesine Yönelik Özellikleri Birleştirme:

C ++

- 1980'de Bjarne Stroustrup tarafından Bell Laboratuvarlarında geliştirildi.
- C ve SIMULA 67'den geliştirildi.
- SIMULA 67'den alınan nesne yönelimli programlama özellikleri
- Hem prosedürel hem de OO programlamayı destekler
- OOP ile birlikte hızla popülerlik kazandı
- ANSI standardı Kasım 1997'de onaylandı
- Çoklu Miras, Kuvvetli tip ayrımı, **dinamik bellek yönetimi**, hazır şablonlara sahip olma ve çok biçimlilik (polymorphism): function ve operator overloading özellikleri vardır.
- Microsoft'un sürümü: MC++ Özellikler, delegeler, arayüzler, çoklu miras yok (Properties, delegates, interfaces, no multiple inheritance)
- En yaygın kullanılan programlama dillerinde biridir. (Özellikle oyun geliştirmede hala kullanılıyor.)
- Kötü yönleri:
 - Büyük ve kompleks bir dil
 - Ada ve Java'dan daha az güvenli

```
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```



Bjarne Stroustrup
**"A History of C++:
1979-1991", 1993**

Objective -C, Swift

```
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        // insert code here...
        NSLog(@"Hello, World!");
    }
    return 0;
}
```

- ◎ Brad Cox ve Tom Love tarafından 1980'lerin sonlarında geliştirilmiştir.
- ◎ Objective-C, C programlama diline Smalltalk tarzı mesajlaşma ekleyen genel amaçlı, nesne yönelimli bir programlama dilidir.
- ◎ Swift'in 2014 yılında piyasaya sürülmesine kadar, Apple tarafından macOS, iOS ve ilgili uygulama programlama arabirimleri (API'lerde) ana programlama diliydi.

Eiffel (1992)

- ◎ Bertrand Meyer tarafından geliştirilmiştir.
- ◎ Emir esaslı ve OO (Object Oriented) özellikleri birleştiren Hybrid bir programlama dilidir.
- ◎ Soyut veri yapılarını, kalıtımı ve dinamik bildirimleri destekler.
- ◎ Altprogram ve çağırıcı (caller) arasındaki iletişim için bildirimler (assertions) kullanır.



```
class
  HELLO_WORLD
create
  make
feature
  make
    do
      print ("Hello, world!")
    end
end
```

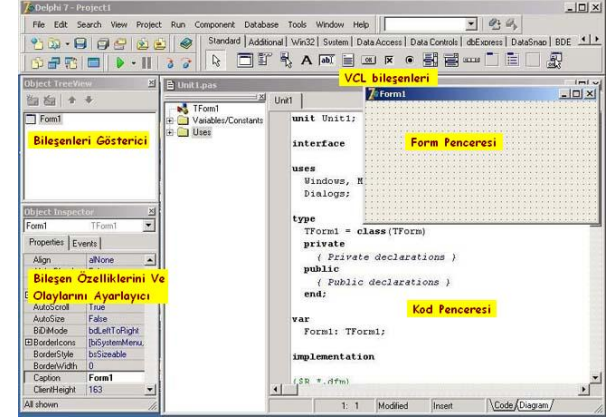
Delphi (1995)

- ◎ Delphi, daha önce Turbo Pascal sistemini geliştiren Anders Hejlsberg tarafından tasarlandı. 1996'da Microsoft'a geçen Hejlsberg, C #'ın baş tasarımcısıydı.
- ◎ Emir esaslı ve OO (Object Oriented) özellikleri birleştiren Hybrid bir programlama dilidir. (C++ ve Objective-C benziyor, fakat Pascal programlama sözdizimi kullanıyor.)
- ◎ Delphi, C ++ 'dan daha zarif ve daha güvenlidir.
- ◎ Delphi ayrıca C ++'dan daha az karmaşıktır. Başka bir deyişle, daha küçük ve basittir. Fakat ifade edilebilirliği ve yazılabilirliği neredeyse eşittir.
- ◎ Delphi, C ++ 'nın bir parçası olan operatorlerin aşırı yüklemesini, generic alt programları ve parametrelili sınıfları içermez.



Delphi (1995)

- Emir esaslı, OO ve hızlı uygulama geliştirme (rapid application development - RAD) özelliklerini başarılı bir biçimde birleştirilmesidir.
- GUI (Graphical User Interface) çok gelişmiş bir programlama diliydi.
- Pascal'dan türemiştir. Bu yüzden, dizi elemanlarının kontrolünde, pointer aritmetiği ve tip zorlamalarında C ve C++'tan daha güvenlidir.
- C++ 'dan daha az komplekstir.
- Kullanıcı tanımlı opertörlere, generic altprogramlara ve parametrize edilmiş sınıflara izin vermez.
- Özellikle Delphi 7 versiyonu günümüze kadar kullanılmıştır.
- Delphi 8 ile .Net teknolojileri yönelmiştir.
- Uzun yıllar Borland firması tarafından geliştirildikten sonra Embarcadero firmasına satılmıştır.



Emir Esaslı Nesneye Yönelik Bir Dil: Java

- ◎ 1990'ların başında Sun'da geliştirildi
 - James Gosling Java tasarım takımına öncülük etmiştir.
 - C ve C ++, **gömülü elektronik cihazlar** için tatmin edici değildi
 - C++ kompleks ve karışıktı
- ◎ Java, basit, taşınabilir ve nesneye yönelik özellikte bir dildir.
- ◎ Miras alma, çok biçimlilik, kuvvetli tip kontrolü, eş zamanlılık kontrolü, dinamik olarak yüklenebilen kütüphaneler, diziler, string işlemleri ve standart kütüphane gibi özellikleri vardır.
- ◎ Bir Java programının temel yapısal bileşeni sınıftır. Bütün veri ve metotlar bir sınıf ile ilişkilidir. Global veri yada fonksiyon yoktur.



Sun Microsystems, 1997

Java

```
class Simple{  
    public static void main(String args[]){  
        System.out.println("Hello Java");  
    }  
}
```

◎ C++ tabanlı sistemlere

- Önemli ölçüde basitleştirilmiş (struct, union, enum, işaretçi aritmetiği ve C ++ atama zorlamalarının yarısını içermez)
- Yalnızca OOP'yi destekler
- Referansları var ama işaretçileri (pointer) yok (Bu özellik güvenlik sağlar)
- Applet'ler için destek ve bir çeşit eşzamanlılık içerir. (Tarayıcılar, applet desteğini geri çekti ve çoğu şu an Java'yı yorumlamıyorlar. Ama web'in gelişmesinde önemli katkıları olmuştur.)
- C++'ta bulunan çoklu miras alma, operatörlerin aşırı yüklenmesi (operator overloading), ve makro ön işleyicisi özellikleri Java'da yoktur.
- Java'da şablon yapıları yoktur. İhtiyaç da minimuma inmiştir.

Java

- ◎ Pointer yoktur. Fakat bütün nesne sınıfları object adlı kök sınıftan miras almaktadır.
- ◎ Thread'lere sahip olduğundan eşzamanlılığı yönetmek kolaydır.
- ◎ **Çöp toplama (Garbage collection)** nesneler için belleği en iyi kullanımı sağlar.
- ◎ Tip dönüşümü kuvvetlidir.
- ◎ Java'da hem derleme hem de yorumlama işlemi vardır. (JIT tabanlı – geçen haftaki derse bakın)
- ◎ Taşınabilir: Java Virtual Machine konsepti, JIT derleyicileri (Geçen hafta anlatılmıştı.)
- ◎ Javascript'in güçlenmesi ile Java istemci tarafında desteği azalmıştır. Çoğu tarayıcı desteklemez.

Betik Programlama Dilleri (Scripting Programming Languages): Perl



Practical Extraction and Report Language



Larry Wall tarafından tasarlandı - 1987'de rapor işlemeyi kolaylaştırmak için genel amaçlı bir Unix betik dili olarak geliştirilmiştir.



Perl yüksek seviyeli, genel amaçlı, **yorumlamalı**, dinamik programlama dillerinden oluşan bir ailedir.



Larry Wall Perl'i yazarken **C**, **sed**, **AWK** ve **ksh** gibi pek çok dilden önemli ve güçlü özellikler ödünç almıştır.



- ksh, yaygın olarak kullanılan ve güçlü öncelikli programlama dillerinden biridir.



- AWK ise rapor üretme dili olarak ortaya çıksa da daha genel bir dile dönüşmüştür.



Web'de CGI programlaması için yaygın kullanım kazandı.



Ayrıca, Unix sistem yönetim dili olarak kullanılır.



Değişkenler: statik tipli ve implicitly



Array'ler dinamiktir. (Çoğu dilde statikti.)



İlişkilendirilebilir diziler (Associative arrays veya hashes): daha hızlı erişilebilir ama bellekte daha fazla yer kaplar.

```
#!/usr/bin/perl
```

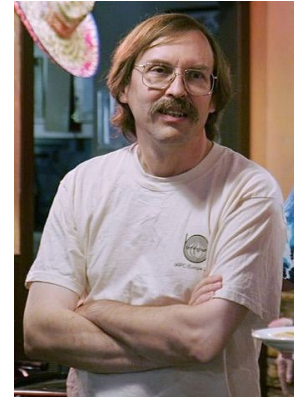
```
# Modules used
```

```
use strict;
```

```
use warnings;
```

```
# Print function
```

```
print("Hello World\n");
```



Nasa'da sistem yöneticisi

<https://tr.wikipedia.org/wiki/Perl>

<https://en.wikipedia.org/wiki/Perl>

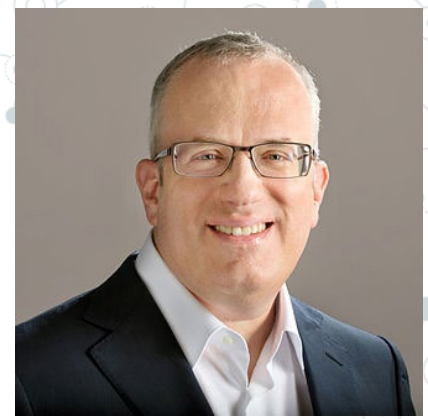
Betik Programlama Dilleri (Scripting Programming Languages): Perl

```
# Perl Example Program
# Input:  An integer, $listlen, where $listlen is less
#         than 100, followed by $listlen-integer values.
# Output: The number of input values that are greater than
#         the average of all input values.
($sum, $result) = (0, 0);
$listlen = <STDIN>;
if (($listlen > 0) && ($listlen < 100)) {
# Read input into an array and compute the sum
  for ($counter = 0; $counter < $listlen; $counter++) {
    $intlist[$counter] = <STDIN>;
  } #- end of for (counter ...)
# Compute the average
  $average = $sum / $listlen;
# Count the input values that are > average
  foreach $num (@intlist) {
    if ($num > $average) { $result++; }
  } #- end of foreach $num ...
# Print result
  print "Number of values > average is: $result \n";
} #- end of if (($listlen ...
else {
  print "Error--input list length is not legal \n";
}
```

Betik Programlama Dilleri:

JavaScript

- Netscape ve Sun Microsystems 'in ortak çalışmaları sonucu 1995'de üretilmiştir. Orijinal adı "LiveScript"tir.
- Yaratıcısı Brendan Eich'tir.
- Yüksek seviyeli JavaScript olaya dayalı (event-driven), fonksiyonel ve emir esaslı programlama stillerini destekler.
- HTML ve CSS ile birlikte çekirdek web teknolojilerinin biridir.
- HTML'i işleyecek ve kontrol edecek çeşitli olanaklar vardır.
- Tarayıcılar içinde Javascript kodlarını için yorumlayıcılar vardır.
- Node.js (2010) ile sunucu tarafında da kullanılmaya başlamıştır.
- TypeScript(2012) Microsoft tarafından geliştirilmekte ve desteklenmekte olan TypeScript; bünyesinde barındırdığı derleyici sayesinde, yazılan kodu JavaScript koduna çevirir. TypeScript gerek istemci taraflı, gerekse sunucu taraflı yazılım geliştirmede kullanılabilir.



```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    alert( 'Hello, world!' );
  </script>

  <p>...After the script.</p>

</body>

</html>
```

Javascript kod örneği

```
// example.js
//   Input:  An integer, listLen, where listLen is less
//           than 100, followed by listLen-numeric values
//   Output:  The number of input values that are greater
//           than the average of all input values
var intList = new Array(99);
var listLen, counter, sum = 0, result = 0;
listLen = prompt (
    "Please type the length of the input list", "");
if ((listLen > 0) && (listLen < 100)) {
    // Get the input and compute its sum
    for (counter = 0; counter < listLen; counter++) {
        intList[counter] = prompt (
            "Please type the next number", "");
        sum += parseInt(intList[counter]);
    }
    // Compute the average
    average = sum / listLen;
    // Count the input values that are > average
    for (counter = 0; counter < listLen; counter++)
        if (intList[counter] > average) result++;
    // Display the results
    document.write("Number of values > average is: ",
        result, "<br />");
} else
    document.write(
        "Error - input list length is not legal <br />");
```

Betik Programlama Dilleri

PHP

- ⦿ PHP: Hypertext Preprocessor, 1994 yılında Rasmus Lerdorf tarafından tasarlanmıştır.
- ⦿ Genellikle Web üzerinden form işleme ve veritabanı erişimi için kullanılan, sunucu tarafında HTML içine gömülmüş betik kodlardır
- ⦿ Sunucu tarafında çalışır
- ⦿ Yorumlamalı – PHP yorumlayıcısı kurulu olmalıdır
- ⦿ Sonradan OOP özellikleri eklenmiştir.
- ⦿ Açık kaynak kodlu olmasının yanı sıra PHP ile kodlanmış birçok açık kaynak kod yazılım internette paylaşılmaktadır.
- ⦿ Değişken mantığı Perl'e benziyor.
- ⦿ PHP Array'leri, Javascript array'leri ve Perl'ün hash'lerini birleştiren bir yapıdadır.



```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```


Betik Programlama Dilleri

Python

- İlk versiyonu geliştiricisi Guido van Rossum tarafından 1991 yılında geliştirilmiştir.
- Python, yorumlamalı, yüksek seviyeli ve genel amaçlı bir programlama dilidir.
- Python'un tasarım felsefesi, kayda değer girinti kullanımıyla kod okunabilirliğini vurgular.
- Dil yapıları ve nesne yönelimli yaklaşımı, programcıların küçük ve büyük ölçekli projeler için net, mantıksal kod yazmalarına yardımcı olmayı amaçlamaktadır.
- Prosedürel (Yapısal), nesneye yönelik ve fonksiyonel programlama yaklaşımlarını destekler.
- Açık kaynak kodludur.
- Üzerine geliştirilen kütüphaneler açık kaynak kod olarak paylaşıldı için birçok projede üretkenliği arttırmak için kullanılır.

```
print('Hello, world!')
```



Betik Programlama Dilleri

Python

- ◎ Dinamik tipli
- ◎ Array yerine list ve tuple, hash yerine dictionary veri tipi vardır.
 - List'ler insert, append, remove ve sort gibi komutları vardır.
- ◎ Concurrency ve thread
- ◎ Soketleri üzerinden ağ programlama
- ◎ Fonksiyonel programlama desteği var
- ◎ Diğer derlenmiş dillerde yazılan uzantıları kullanabilir (ilginç özelliklerinden biri)

```
print('Hello, world!')
```



Betik Programlama Dilleri

Ruby

```
puts "Hello World!"
```

- ◎ Ruby, yorumlanmalı, yüksek seviyeli, genel amaçlı bir programlama dilidir.
- ◎ Japonya'da Yukihiro "Matz" Matsumoto tarafından 1990'ların ortasında tasarlanmış ve geliştirilmiştir.
- ◎ Yaratıcısı Perl, Smalltalk, Eiffel, Ada, BASIC ve Lisp'ten etkilenmiştir.
- ◎ Prosedürel, nesneye yönelik ve fonksiyonel programlama yaklaşımlarını destekler.
- ◎ Perl ve Python'un yerini almaya başladı
- ◎ Saf bir nesne yönelimli betik dili (Smalltalk gibi)
 - Tüm veriler nesnelerdir
- ◎ Sözdizimi Eiffel ve Ada'ya benzer: dinamik tip
- ◎ Çoğu operatör, kullanıcı kodu ile yeniden tanımlanabilen yöntemler olarak uygulanır.
- ◎ Japonya'nın ilk tasarlanan dili olmasına rağmen Amerika'da geniş bir kullanıma erişmiştir.



Betik Programlama Dilleri

Lua

```
print("Hello World!")
```

- © 1993'te Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes tarafından yaratılmıştır.
- © Nesneye yönelik yorumlamalı bir betik dilidir.
- © Ağırlıklı olarak gömülü sistemler ve istemciler için tasarlanmış hafif (lightweight) ve yüksek seviyeli bir programlama dilidir.
- © Lua çapraz platformdur, çünkü derlenen bayt kodunun yorumlayıcısı C ile yazılmıştır ve Lua bunu uygulamalara yerleştirmek için nispeten basit bir C API'ye sahiptir.
- © List, tuple, hash gibi veri tiplerini table isminde tek veri tipinde birleştirir.
- © Lua tasarımcıları hız, taşınabilirlik, genişletilebilirlik ve geliştirme sırasında kullanım kolaylığını iyileştirmeye odaklandı.

.Net'in Amiral Gemisi Dili: C#

- .Net geliştirme platformunun bir parçası olarak tasarlandı. (2000)
- İlk tasarımcısı Anders Hejlsberg'tir. (Delphi)
- C++, Java ve Delphi gibi dillerin özelliklerini kullanır.
- .Net geliştirme ortamı C#, VB.Net, C++, F# ve vb. birçok programlama diline imkan tanır. C#, .Net'in en önemli dilidir.
- Pointer, delegate, property, enumeration types, sınırlı dinamik tiplendirme, anonymous types içerir.
- JIT kullanıyor.
- Operator overloading özelliği vardır.
- enum, struct ve switch gibi bilinen özellikleri daha iyi hale getirmiştir. Örneğin enum özelliği C++ çok güvenli değilken, C#'ta daha güvenli hale getirilmiştir.
- Gelişmeye devam ediyor... (Geliştirme ortamı olarak ta .Net Core, .Net 5 ...)

```
using System;
class HelloWorld
{
    public static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

C# kod örneği

```
// C# Example Program
// Input: An integer, listlen, where listlen is less than
//        100, followed by listlen-integer values.
// Output: The number of input values that are greater
//         than the average of all input values.
using System;
public class Ch2example {
    static void Main() {
        int[] intlist;

        int listlen,
        counter,
        sum = 0,
        average,
        result = 0;
        intList = new int[99];
        listlen = Int32.Parse(Console.ReadLine());
        if ((listlen > 0) && (listlen < 100)) {
            // Read input into an array and compute the sum
            for (counter = 0; counter < listlen; counter++) {
                intList[counter] =
                    Int32.Parse(Console.ReadLine());
                sum += intList[counter];
            } //- end of for (counter ...)
            // Compute the average
            average = sum / listlen;
            // Count the input values that are > average
            foreach (int num in intList)
                if (num > average) result++;
            // Print result
            Console.WriteLine(
                "Number of values > average is:" + result);
        } //- end of if ((listlen ...
        else
            Console.WriteLine(
                "Error--input list length is not legal");
    } //- end of method Main
} //- end of class Ch2example
```

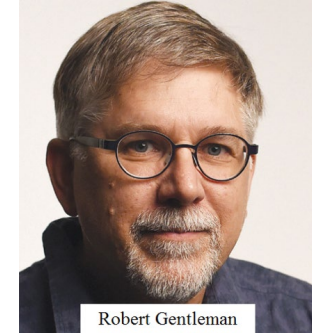
Performansı Ön Plana Çıkaran Dil: Rust

- ⦿ Rust, orijinal olarak Dave Herman, Brendan Eich ve diğerlerinin katkılarıyla Mozilla Research'te Graydon Hoare tarafından tasarlandı ve ilk 2010 yılında duyuruldu.
- ⦿ Rust, performans ve güvenlik, özellikle de güvenli eşzamanlılık için tasarlanmış çok paradigmatlı (prosedürel, nesneye yönelik ve fonksiyonel) bir programlama dilidir.
- ⦿ Bellek güvenliğini garanti eder. (C++'a göre en büyük farkı)

```
fn main() {  
    println!("Hello, World!");  
}
```

İstatistiksel hesaplama: R Programlama Dili

- © R, istatistiksel hesaplama ve grafikler için yazılım ortamı olup aynı zamanda programlama dilidir.
- © Yeni Zelanda Auckland Üniversitesinden Ross Ihaka ve Robert Gentleman tarafından 1993'te ortaya çıkarılan R, hâlihazırda R Geliştirme Çekirdek Ekibi tarafından geliştirilmektedir. S programlama diline benzeyen R, S'nin uyarlaması olarak değerlendirilebilir.
- © İstatistiki yazılım geliştirme için istatistikçiler arasında de facto standart haline gelen R, istatistiki yazılım geliştirme ve veri analizi alanında kullanılmaktadır



```
cat("Hello world\n")
```

[https://tr.wikipedia.org/wiki/R_\(programlama_dili\)](https://tr.wikipedia.org/wiki/R_(programlama_dili))

Diğer Programlama Dilleri

- ◎ Dart: birden çok platformdaki uygulamalar için istemci için optimize edilmiş bir programlama dilidir. Google tarafından geliştirilmiştir ve mobil, masaüstü, sunucu ve web uygulamaları oluşturmak için kullanılır. (2011)
- ◎ Go, Google'da tarafında geliştirilen performansa ve düşük gecikmeye odaklanan bir dildir. (2009)
- ◎ Swift, Apple tarafından iOS ve macOS platformlarına iOS ve Mac uygulamaları geliştirmek için oluşturulan, derlenerek çalışan güçlü ve kullanımı kolay, nesne yönelimli bir programlama dilidir. (2014)

Dillerin son durumu: Ranking of Programming Languages

- © [index | TIOBE - The Software Quality Company](#)
- © [PYPL PopularitY of Programming Language index](#)
(Google search)
- © [Metabase \(intellimenta.com\)](#) (Stack Overflow)
- © [Top Computer Languages 2020 - StatisticsTimes.com](#) (Hepsini kıyaslayan...)