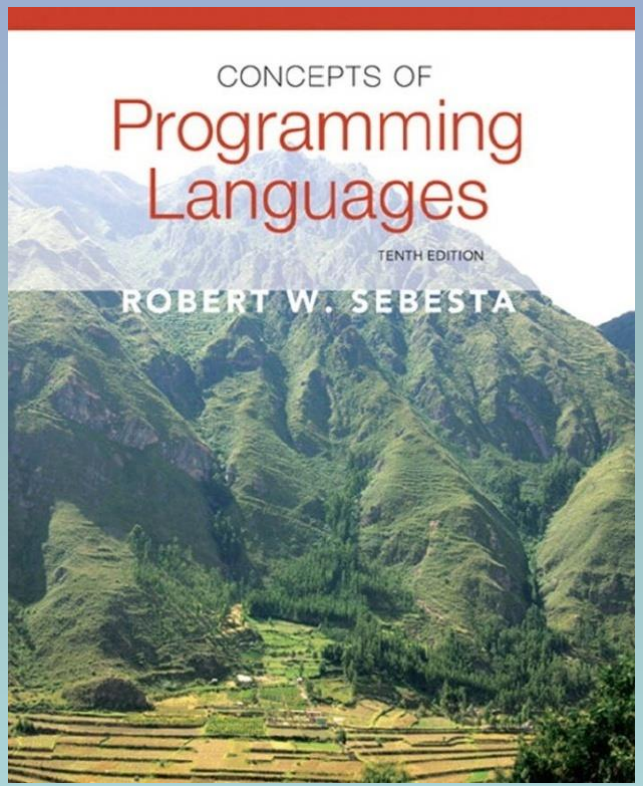
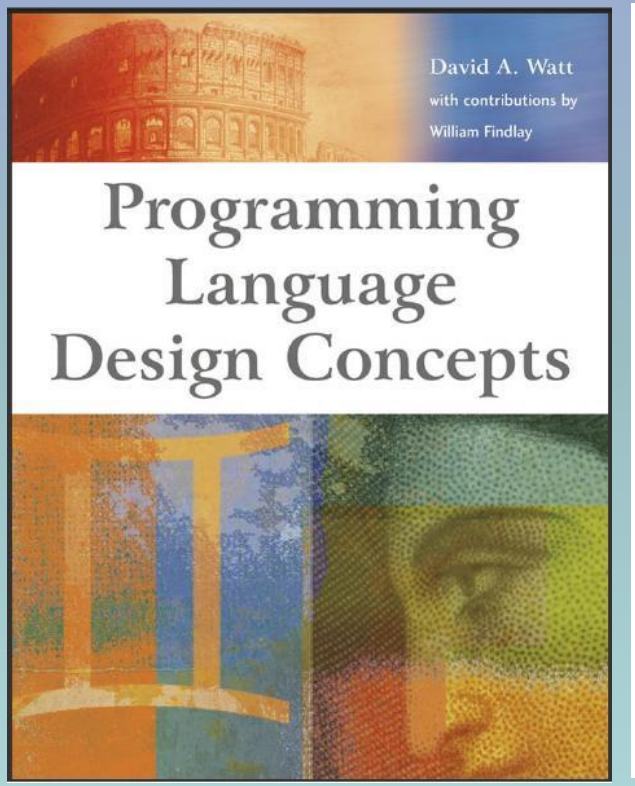
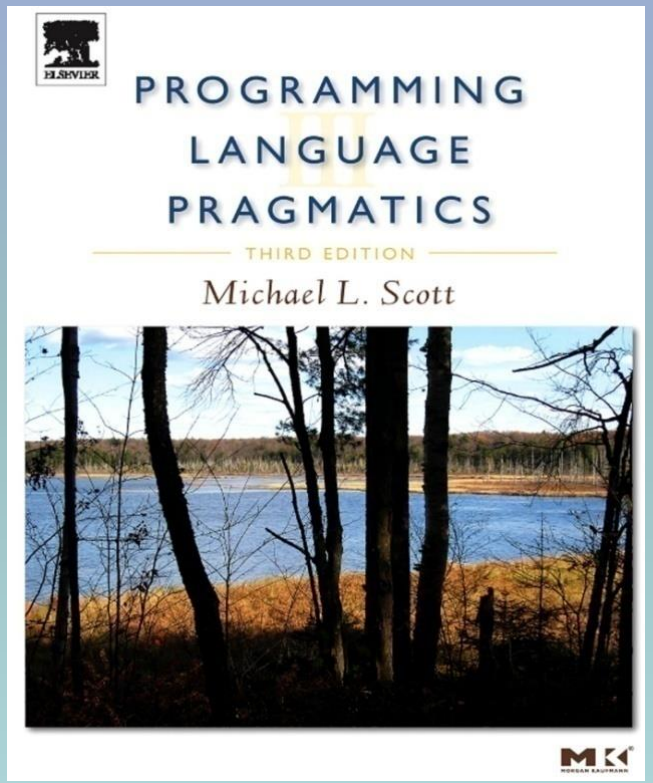
Bölüm 2: Önemli Programlama Dillerinin Gelişimi







Bölüm 2 Konular

1. Zuse'nin Plankalkul' ü
2. Minimum Donanım Programlama: Sözde kod (Pseudocode)
3. IBM 704 ve Fortran
4. Fonksiyonel Programlama: LISP
5. Sofistikeliğe doğru ilk adım: ALGOL 60
6. Ticari Kayıtları bilgisayara uyarlamak: COBOL
7. Zaman Paylaşımının (Timesharing) başlangıcı: BASIC

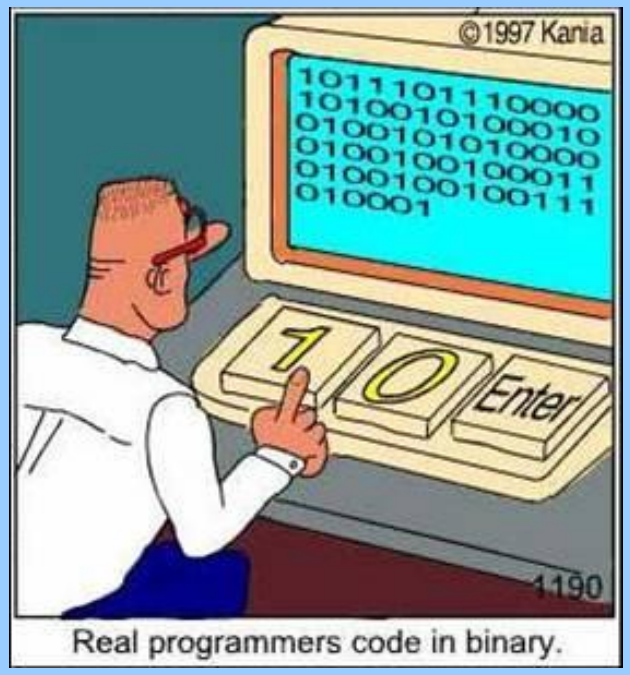
Bölüm 2 Konular (devamı)

8. Herkes için Herşey: PL/I
9. İlk iki Dinamik Dil: APL ve SNOBOL
10. Veri Soyutlama (Data Abstraction) nın başlangıçları: SIMULA 67
11. Ortogonal (Orthogonal) Dizayn: ALGOL 68
12. ALGOL'lerin ilk torunlarından bazıları
13. Mantık (Logic) temelli programlama: Prolog
14. Tarihin en büyük tasarım çabası: Ada

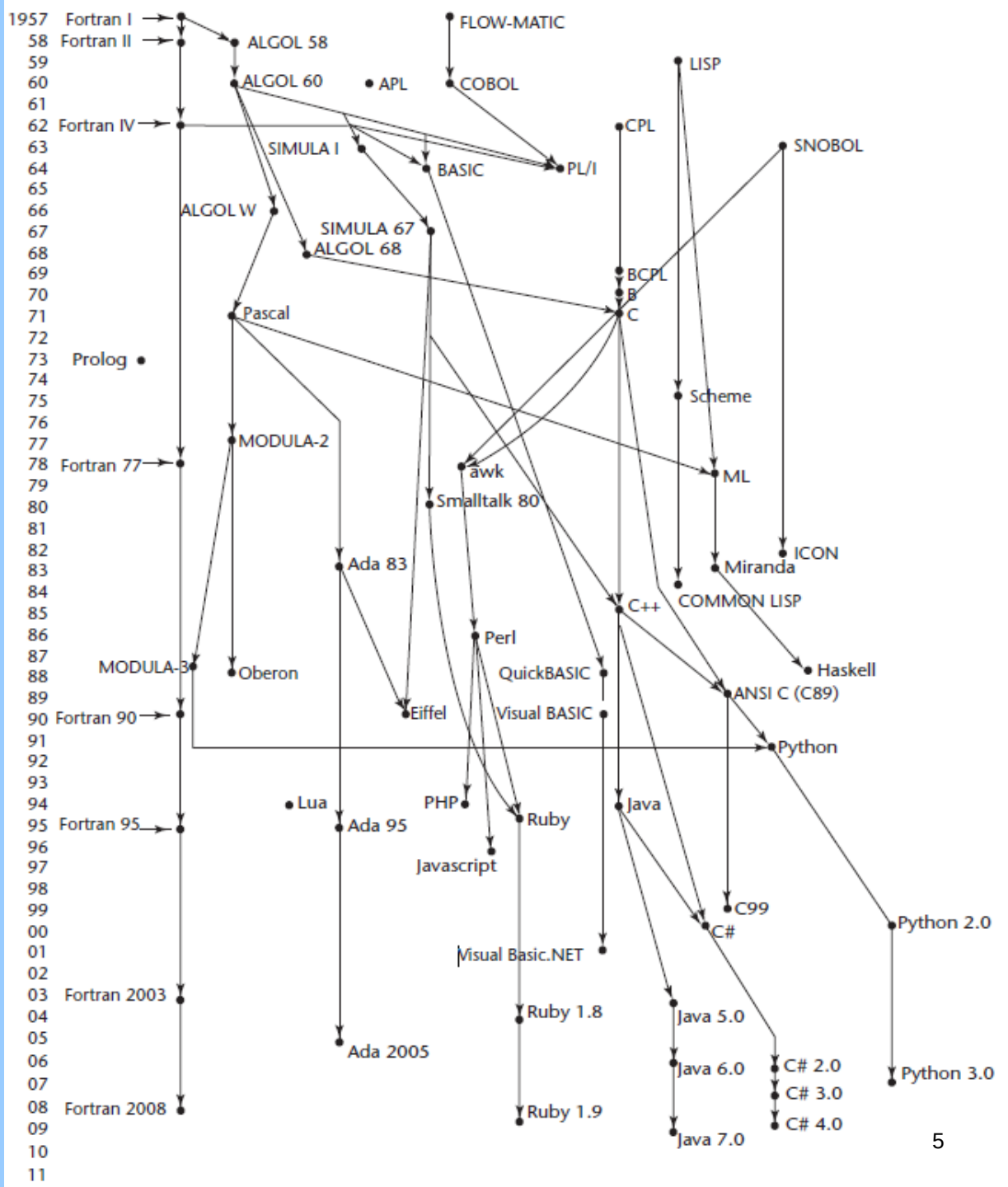
Bölüm 2 Konular (devamı)

15. Nesneye-dayalı Programlama: Smalltalk
16. Zorunlu(Imperative) ve nesneye-dayalı (Object-Oriented) özellikleri birleştirmek: C++
17. Bir Zorunlu nesneye-dayalı dil (Imperative-Based Object-Oriented): Java
18. Betik Diller (Scripting Languages): JavaScript, PHP, ve Python
19. Yeni milenyum için C-temelli bir dil: C#
20. İşaretleme (Markup)/Programlama Hibrit Diller

Yaygın Dillerin Soyağacı (Genealogy)

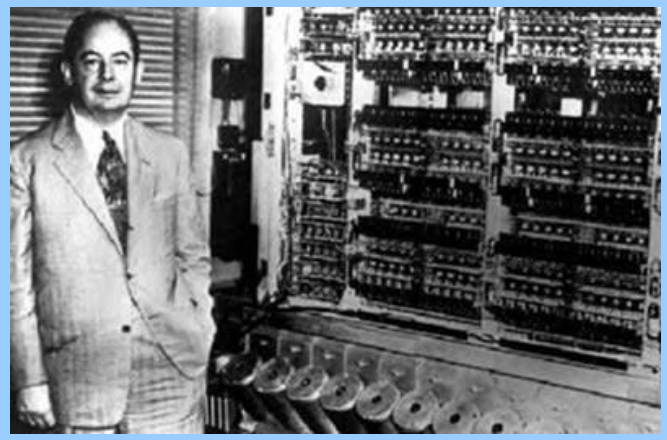


Real programmers code in binary.



ENIAC

İlk programlama dili ilk bilgisayar olan ENIAC (1943) ile beraber gelen ENIAC Programlama Dili'dir. ENIAC sadece matematiksel işlemler yapabilen sadece ikili sayı sistemi (0-1) ile çalışan devrim niteliğinde bir dildi. Bilgisayarın delikli kartları okuması sonucu elde edilen ikili sayı sistemi tabanındaki sayısal veri ile işlemler yapan ENIAC dilinin programlama mantığı hâlen geçerli olan bir mimaridir.



2.1 Zuse'nin Plankalkül'ü

- ENIAC dilinin muazzamlığı karşısında ilk atağa geçen Almanya olmuş ve başarılı da olmuştur. PLANKALKÜL adı verilen Almanya resmi programlama dili olarak nitelendirilebilinecek bir programlama dili Konrad Zuse liderliğinde geliştirilmiş ve istenilen işlemleri ENIAC' a göre daha hızlı yapma başarısını göstermiştir.
- Fakat ENIAC ekibi bu süre zarfında daha avantajlı oldukları için çok hızlı bir yanıt verebilmiş ve PLANKALKÜL dilinin çok fazla duyulmasını engellemiştir.

2.1 Zuse'nin Plankalkül'ü

- İlk yüksek seviye (donanımın ayrıntılarına bağlı olmayan) programlama dili
- Asla geliştirilmedi
- İleri veri yapıları
  - Kayan nokta (floating point), diziler (arrays), kayıtlar (records)
- Sabitler (Invariants)

Konrad Zuse (1910-1995)

Plankalkül Sentaksı

- A[4] + 1 deyimini A[5] ‘e atayan bir ifade

```
      |   A + 1 => A
  V   |   4         5    (altsimgeler-subscripts)
  S   |   1.n       1.n  (veri tipleri-data types)
```

Plankalkül

- PLANKALKÜL sonunda daha kapsamlı bir şekilde 1972 yılında yayımlandı ve bunun için ilk derleyici 1998 yılında hayata geçirildi. Başka bir bağımsız uygulama Berlin Hür Üniversitesi tarafından 2000 yılında izledi. “Kalkül” resmi sistem anlamına gelir – Hilbert tarzı mahsup sistemi başlangıçta “Hilbert-Kalkül” olarak adlandırıldı. Bu yüzden Plankalkül “planlamak için resmi sistem” demektir.
- Şuan geniş kitlilere hitap edebilen veya öğrenilebilir bir dil olmadığı için kullanılan bir dil olmayabilir ama, Almanlar her zamanki gibi taviz vermeyen tavrını korumuş ve resmi dilini yaşatmaya çalışmıştır.

2.2 Minimum Donanım Programlama: Sözdekodlar (Pseudocodes)

- Makine kodu kullanmak neden yanlıştı?
  - Az okunabilirlik
  - Az değiştirilebilirlik
  - Deyim kodlama (Expression coding) usandırıcıydı
  - Makine eksiklikleri—indeksleme veya kayan nokta (floating point) yoktu

Sözdekodlar: Short Code (Kısa Kod)

- Short Code,1949 yılında Mauchly tarafından BINAC bilgisayarları için geliştirildi
  - Deyimler (Expressions) -soldan sağa doğru- kodlandı
  - İşlemlerden örnekler:

```
01 - 06 abs value 1n (n+2)nd power
02 ) 07 +         2n (n+2)nd root
03 = 08 pause     4n if <= n
04 / 09 (         58 print and tab
```

Sözdekodlar: Speedcoding (hızlıkodlama)

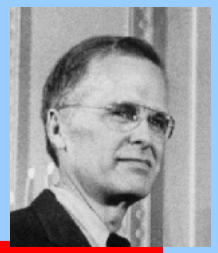
- Speedcoding 1954 yılında Backus tarafından IBM 701 için geliştirildi
- Aritmetik ve matematiksel fonksiyonlar için sözde işlemler
  - Koşullu (conditional) ve koşulsuz (unconditional) dallanma (branching)
  - Dizi erişimi için kaydedicileri (registers) otomatik arttırır
  - Yavaştır!
  - Kullanıcı programı için sadece 700 kelime ayrılmıştır

Sözdekodlar : İlgili Sistemler

Grace Hopper

- UNIVAC Derleme Sistemi
  - Grace Hopper yönetimindeki bir ekip tarafından geliştirilmiştir
  - Sözdekod makine koduna genişletilmiştir
- David J. Wheeler (Cambridge University)
  - Salt Adresleme (absolute addressing) problemini çözmek için yeniden-yerleştirilebilir adres blokları kullanan bir metot geliştirmiştir

2.3 IBM 704 ve Fortran



- **Fortran 0: 1954** – IBM firmasında John Backus tarafından geliştirilmiş ancak uygulanmamıştır
- **Fortran I:1957 – Ticari olarak kullanıma sunuldu**
  - İndeks yazmaçları (registers) ve kayan nokta (floating point) donanımına sahip yeni IBM 704 için tasarlanmıştır
  - Bu durum derlenmiş programlama dilleri fikrine yol açtı, çünkü yorumlama maliyetini saklayacak hiçbir yer yoktu (hiçbir kayan nokta yazılımı)
  - Geliştirme platformu
    - Bilgisayarlar küçük ve güvenilmezdi
    - Uygulamalar bilimseldi
    - Programlama metodolojileri ve araçları yoktu
    - Makine verimliliği en önemli sorundu

Fortran'ın tasarım işlemi

- Fortran I'in tasarımına platformun (environment) etkisi
  - Dinamik belleğe (storage) ihtiyaç yoktu
  - İyi bir dizi (array) işleme ve sayma döngülerine (counting loops) ihtiyaç vardı
  - İş yazılımı (ticari ürünler) için string işleme, ondalık aritmetik, veya güçlü girdi/çıktı yoktu

Fortran I ‘e bakış

- FORTRAN’ın gerçekleştirilmiş ilk sürümü
  - İsimler altı karaktere kadar olabiliyordu
  - Test-sonrası sayma döngüsü (Post-test counting loop) (`DO`)
  - Biçimlendirilmiş Girdi/Çıktı (Formatted I/O)
  - Kullanıcı-tanımlı altprogramlar
  - Üçlü seçim ifadesi (aritmetik `IF`)
  - Veri tipi ifadeleri yoktur

Fortran I ‘e bakış (devamı)

- FORTRAN’ın gerçekleştirilmiş ilk sürümü
  - Ayrı derleme yoktur
  - Derleyici (compiler), 18 iş-yılı çabadan sonra Nisan 1957’de çıktı
  - 400 satırdan fazla programlar, 704’ün az güvenilirliği yüzünden nadiren doğru derleniyordu
  - Kod çok hızlıydı
  - Kısa zamanda yaygın kullanılır hale geldi

Fortran II

- 1958’ de yayıldı
  - Bağımsız derleme
  - Hataları (bugs) düzeltti

Fortran IV

- 1960-62 yıllarında geliştirildi
  - Belirtilmiş (Açık, explicit) tip tanımlamaları
  - Mantıksal seçim ifadesi
  - Altprogram (Subprogram) isimleri parametre olabilir
  - 1966 yılında ANSI standardını aldı

Fortran 77

- 1978 de yeni standart haline geldi
  - Karakter dizisi (string) işleme
  - Mantıksal döngü kontrol ifadesi
  - **`IF-THEN-ELSE`** ifadesi

Fortran 90

- Fortran 77'den en önemli farkları
  - Modüller
  - Dinamik diziler (arrays)
  - İşaretçiler (Pointers) – bağlı liste, dinamik bellek
  - Özyineleme (Recursion)
  - **`CASE`** ifadesi
  - Parametre tipi testi (parameter type checking)
  - Bit düzeyinde işlem
  - Dizi yapıları daha iyi kullanılabiliyor
  - Altprogram yapıları daha esnek,
  - İsimler daha uzun yazılabiliyor (okunabilirlik)

Fortran'ın en son versiyonu

- Fortran 95 – nispeten ufak eklemeler, artı bazı silmeler
  - Pointer ve yapılara varsayılan olarak ilk değer atanması ve taşınabilirliğin mükemmel hale getirilmesi temel hedef olmuştur.
  - Nesneye dayalı programlama özellikleri de FORTRAN dil ailesine sunulmaya çalışılmaktadır.
- Fortran 2003 – aynı

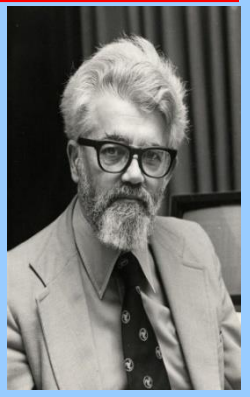
Fortran Değerlendirmesi

- Çok iyi optimize eden derleyiciler (90'dan önceki tüm sürümler)
  - Bütün değişkenlerin tipleri ve bellekleri çalışma zamanından (run-time) önce düzeltilir
- Bilgisayarların kullanılma şeklini sürekli çarpıcı biçimde değiştirdi
- Bilgisayar dünyasının *lingua franca (uluslarası dil)* 'sı (geçerli dili) olarak karakterize edildi
- 1950'li yıllarda makine dilinde yazılan kodların, çevrilmiş kodlara göre daha hızlı çalışmasına rağmen FORTRAN getirdiği kullanım kolaylığı ile büyük bir başarı sağlamıştır.
- Günümüze kadar bir çok sürümü çıkmış ve yenilenmiştir. Sayısal uygulamaların programlanması için uygun bir dildir.

Fortran Örnek

```
Program Example
! Fortran 95 example program
Implicit none
Integer :: Int_List(99)
Integer :: List_Len, Counter, Sum, Average, Result
Result = 0
Sum =0
Read *, List_len
If ((List_Len > 0) .AND. (List_Len <100)) Then
   Do Counter=1 List_Len
      Read *,Int_List(Counter)
      Sum = Sum + Int_List(Counter)
   End Do
   Average = Sum / List_len
   Do Counter=1, List_Len
      If (Int_List(Counter) > Average) Then
         Result = Result + 1
      End If
   End Do
   Print *, 'Number of Values > average is ', Result
Else
   Print *, 'Error - list length value is not legal '
End If
End Program Example
```

2.4 Fonksiyonel Programlama: LISP



John McCarthy

- LISt Processing (Liste işleme)
  - McCarthy tarafından MIT'de IBM 704 bilgisayarlar için tasarlandı
- AI (Artificial Intelligence–yapay zeka) araştırmasının ihtiyaç duyduğu dil şöyleydi:
  - Veriyi liste halinde işleme (dizi (array) yerine)
  - Sembolik hesaplama (sayısal yerine)
- Sadece iki veri tipi: *atom*lar and *list* (e)ler
- Sentaks *lambda calculus*'a dayalıdır

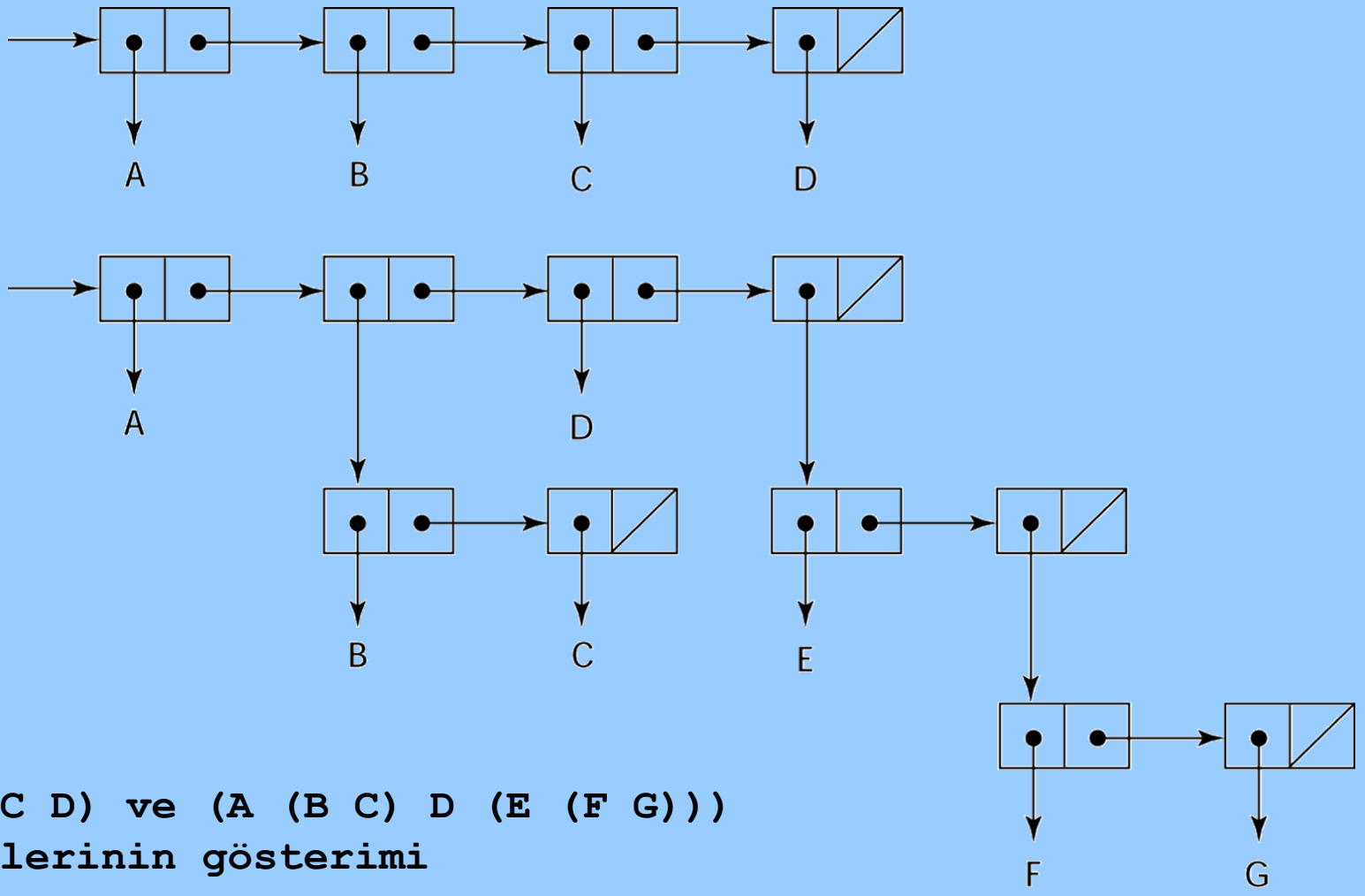
2.4 Fonksiyonel Programlama: LISP

- Program kodu ve veriler tam olarak aynı formdadır:
- Örmek: (A B C D)
  - Bu Eğer bir veri ise , A, B, C ve D verilerini içeren bir listedir
  - Eğer bir kod ise o zaman A fonksiyonunun B, C ve D parametrelerine uygulanması olarak yorumlanır.
- Olağanüstü esneklik, ifade gücü ve kodun aynı zamanda veri olarak da kullanılabilmesi özelliği LISP'i yapay zeka uygulamalarında rakipsiz hale getirmiştir.

2.4 Fonksiyonel Programlama: LISP

- LISP esas olarak yorumlayıcı kullanan bir dildir. Derleyici kullanan versiyonları da vardır.
- Veri tiplemesi bakımından esnek bir dildir.
- Nesne Yönelimli Programlamayı destekler
- Veri tabanlarına erişim ve GUI olanağı vardır.
- Çok iyi belgelendirilmiş olup, COMMON LISP, ANSI tarafından standart hale getirilmiştir.

İki LISP Listesini Gösterimi



(A B C D) ve (A (B C) D (E (F G)))
Listelerinin gösterimi

LISP Değerlendirmesi

- Fonksiyonel programlamada öncü olmuştur
  - Değişkenlere(variables) veya atamaya (assignment) ihtiyaç yoktur
  - Kontrol, özyineleme (recursion) ve koşullu ifadeler (conditional expressions) ile sağlanır
- Halen AI için hakim olan dildir
- Diferansiyel ve integral hesaplamalarında,elektrik devre teorilerinde, sayısal mantık ve yapay zekanın bir çok alanında sembolik hesaplamalar için kullanılmıştır.
- COMMON LISP ve Scheme, LISP'in güncel diyalektleridir (lehçeleridir-versiyonlarıdır)
- ML, Miranda, ve Haskell ilgili dillerdir

LISP Örnek

```
(DEFUN equal_lists (list1 list2)
    (COND
        ((ATOM list1) (EQ list1 list2))
        ((ATOM list2) NIL)
        ((equal_lists (CAR list1) (CAR lsit2))
            (equal_lists (CDR lsit1) (CDR list2)))
        (T NIL)
    )
)
```

Scheme

- MIT ‘de 1970’lerin ortalarında geliştirildi
- Küçüktür
- Statik kapsam (static scoping)’ın geniş kullanımı
- Fonksiyonlar birinci-sınıf varlıklardır (first-class entities)
- Basit sentaksı (ve küçük boyutu), eğitim amaçlı uygulamalar için ideal olmasını sağlamıştır

COMMON LISP

- LISP'in birkaç diyalektinin özelliklerini bir dilde toplama gayretidir
- Büyük, karmaşık

2.5 Sofistikeliğe doğru ilk adım: ALGOL 60

Peter Naur

- ALGOrithmic Language
- FORTRAN I'den esinlenilmiştir
- Geliştirme platformu
  - FORTRAN ancak IBM 70x içindi
  - Geliştirilmekte olan diğer bütün diller belirli makineler içindi
  - Taşınabilir dil yoktu; hepsi makine-bağımlıydı
  - Haberleşme algoritmaları için evrensel bir dil yoktu
- ALGOL 60 evrensel bir dil tasarlama çabalarının sonucuydu

İlk tasarım işlemi

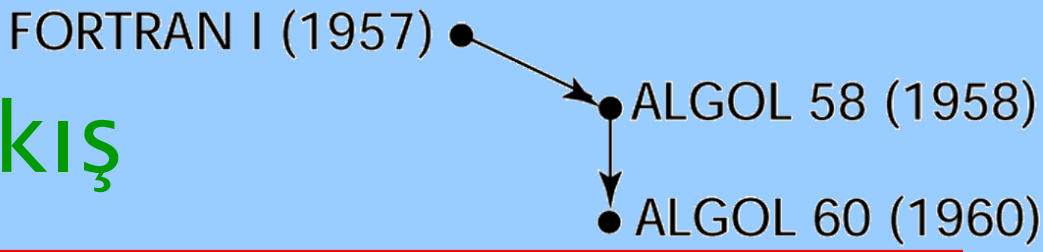
- ACM ve GAMM tasarım için Zürich'te dört gün görüştü (Mayıs 27 den Haziran 1'e, 1958)
  - FORTRAN dilinin başarısını gören GAMM (German Society of Applied Mathematics) ve ABD'de bulunan ACM (Association for Computer Machinery) Peter Naur başkanlığında ortak bir komite ile International Algorithmic Language (IAL) adı verilen dili geliştirdiler.
- Dilin amaçları
  - Matematiksel gösterime yakın
  - Algoritma tanımlamak için iyi
  - Makine koduna çevrilebilir olmalıydı

ALGOL 58

- Tip kavramı resmileştirildi
- İsimler herhangi bir uzunlukta olabilirdi
- Diziler (arrays) herhangi bir sayıda altsimgeye(subscripts) sahip olabilirdi
- Parametreler kip (mode) ile ayrıldı (in & out)
- Altsimgeler (Subscripts) köşeli parantezler içine yerleştirilmişti
- Bileşik (compound) ifadeler (**`begin ... end`**)
- Noktalı virgül ayırıcı olarak kullanıldı
- Atama ifadesi olarak ilke evrensel kullanım --- işleci := oldu (*variable := expression*)
- **`if`** 'in **`else-if`** deyimi vardı
- I/O yoktu – "onu makine bağımlı hale getirirdi"

ALGOL 58 Implementasyonu

- Geliştirmesi planlanmadı, fakat çeşitleri şunlardı: (MAD, JOVIAL)
- Başlangıçta IBM istekli olmasına rağmen, 1959 ortalarında tüm destek geri çekildi



ALGOL 60 'e bakış

- Paris'teki 6-günlük toplantı sonucunda ALGOL 58'in değiştirilmesiyle geliştirildi
- Yeni özellikler
  - Blok yapısı (yerel kapsam--local scope)
  - İki parametre geçişi (parameter passing) metodu (**pass by value ve pass by name**)
  - Altprogram özyineleme (recursion)
  - Yığın-dinamik diziler (Stack-dynamic arrays)
  - Hala I/O (girdi/çıktı) ve dizim (string) işleme yoktu

ALGOL 60 Değerlendirmesi

- Başarıları
  - 20 yılı aşkın süre algoritma yayınlamanın standart yolu olarak kalmıştır
  - Sonra gelen bütün zorunlu (imperative) diller ona dayandırılmıştır
  - Fakat programlama dili tasarımında **blok kavramı**, **alt programlar**, **özyineleme**, **yığın** ve **diziler** gibi bir çok yeniliğe öncülük etmiştir.
  - İlk makine-bağımsız dildir
  - Sentaksı resmi olarak tanımlanan ilk dildir (BNF)
  - Rekürsiyon ve blok yapısını desteklemek için donanımda yığıta izin verildiğinden bilgisayar mimarisini etkilemiştir

ALGOL 60 Değerlendirmesi (devamı)

- Başarısızlıkları
  - Hiçbir zaman, özellikle U.S.'de yaygın olarak kullanılmamıştır
  - 60 ve 70'li yıllar arasında akademik olarak çok kullanılmasına rağmen ticari alanda başarı elde edememiş bir dildir.
  - Nedenleri
    - I/O ve karakter kümesinin eksikliği programları taşınamaz yapmıştır
    - Aşırı esnek—geliştirilmesi zor (Anlaşılır olmakta zorluk ve yürütmede (implementation) verimsizlik)
    - Fortran'ın köklü olması ve sağlamlaştırılması
    - Resmi sentaks tanımı
    - O zamanlar bilişim sektörünün %80'ine sahip IBM desteğinin eksikliği (Çünkü bu sırada IBM FORTRAN dilinde yazılmış zengin bir kütüphaneye sahiptir ve haliyle FORTRAN'ı desteklemektedir.)

ALGOL Örnek

```
// the main program (this is a comment)
BEGIN
   FILE F (KIND=REMOTE);
   EBCDIC ARRAY E [0:11];
   REPLACE E BY "HELLO WORLD!";
   WHILE TRUE DO
   BEGIN
      WRITE (F, *, E);
   END;
END.
```

```
// the main program (this is a comment)
begin
   integer N;
   Read Int(N);
   begin
      real array Data[1:N];
      real sum, avg;
      integer i; sum:=0;
      for i:=1 step 1 until N do
      begin
           real val;
           Read Real(val);
           Data[i]:=if val<0 then -val else val
      end;
      for i:=1 step 1 until N do
          sum:=sum + Data[i];
      avg:=sum/N;
      Print Real(avg)
   end
end
```

Ticari Kayıtları bilgisayara uyarlamak: COBOL

- COmmon Business Oriented Language)
- Geliştirme platformu
  - UNIVAC, FLOW-MATIC'i kullanmaya başlıyordu
  - USAF, AIMACO' yu kullanmaya başlıyordu
  - IBM , COMTRAN'ı geliştiriyordu

FLOW-MATIC (1957)

COBOL (1960)

Grace Hopper

COBOL Tarihi arkaplan

- İngilizce sözcük yada cümle yapılarını kullanan ve İŞLETMELERe yönelik ortak bir dildir.
- Özellikle bilgisayara büyük miktarda bilgi giriş-çıkışının yapıldığı uygulamalar için geliştirilmiştir (Stok kontrol, Bordro)
- FLOW-MATIC temellidir
- FLOW-MATIC özellikleri
  - İsimler gömülü tirelerle (kısa çizgi- hyphen) 12 karaktere kadar çıkabiliyordu
  - Aritmetik operatörler için İngilizce isimler (aritmetik deyimler yoktu)
  - Veri ve kod tamamen ayrıydı
  - Her ifadede fiil (verb) ilk kelimeydi

COBOL Tasarım İşlemi

- İlk tasarım toplantısı (Pentagon) – Mayıs 1959
- Tasarım amaçları
  - Basit İngilizce gibi görünmeli
  - Daha az güçlü olacağı anlamına gelse bile kullanımı kolay olmalı
  - Bilgisayar kullanıcıların tabanını genişletmeli
  - Mevcut derleyici problemleriyle kısıtlanmış olmamalı
- Tasarım komitesi üyelerinin tamamı bilgisayar üreticilerinden ve DoD (Dept. Of Defence – Amerikan savunma bakanlığı) birimlerinden oluşuyordu
- Tasarım Problemleri: Aritmetik ifadeler? Altsimgeler (indisler, subscripts)? Üreticiler arasında kavgalar


CCSDL

COBOL Değerlendirmesi

- Katkılar
  - Bir yüksek-düzeyli dilde ilk kez makro olanağı
  - Hiyerarşik veri yapıları (records) ve yan anlamlı isimlerin (**connotative names)** görüldüğü ilk dillerdir
  - İç içe (nested) seçim ifadeleri
  - Uzun isimler (30 karaktere kadar), tirelerle birlikte
  - Ayrı veri bölümü (data division)
    - Program içerisindeki dil ifadeleri ve fiziksel adresleri verilerin tanımlandığı ve çalıştırılabilir işlemlerin bulunduğu yer olarak ikiye ayrılır.
  - Özellikle raporlama açısından çeşitli olanaklar sağlamaktadır
  - Yoğun miktarda verileri işlemede kolaylık sağlayan deyimleri ve yapıları sayesinde ticari uygulamalarda çok popüler olmuştur.

COBOL

- COBOL içerisinde rapor yazdırma, tablo içinde arama yapma, ve kullanımı çok kolay olan dosya sıralama rutinleri bulunmaktadır.
- Yapısal özellikleri dolayısıyla veritabanı yönetim sistemleri ile birlikte kullanımı kolaydır.

- Fonksiyonları desteklememektedir

COBOL: DoD Etkisi

- DoD tarafından ihtiyaç duyulan ilk dil
  - DoD olmasaydı başarısız olacaktı
- Halen yaygın kullanılan ticari uygulama dilidir
- 1990'larda ise OOP versiyonu üretilmiştir.
- Üzerinde hala çalışmalar yapılmakta ve Amerika hala büyük miktardaki verileri işlerken COBOL kullanmaktadır

COBOL Örnek

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300 DATE-WRITTEN. 02/05/96 21:04.
000400* AUTHOR BRIAN COLLINS
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200 100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500 DISPLAY "HELLO, WORLD." LINE 15 POSITION 10.
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SeqWrite.
AUTHOR.  Michael Coughlan.
* Example program showing how to create a sequential file
* using the ACCEPT and the WRITE verbs.
* Note: In this version of COBOL pressing the Carriage Return (CR)
* without entering any data results in StudentDetails being filled
* with spaces.


ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT StudentFile ASSIGN TO "STUDENTS.DAT"
                ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD StudentFile.
01 StudentDetails.
   02  StudentId       PIC 9(7).
   02  StudentName.
       03 Surname      PIC X(8).
       03 Initials     PIC XX.
   02  DateOfBirth.
       03 YOBirth      PIC 9(4).
       03 MOBirth      PIC 9(2).
       03 DOBirth      PIC 9(2).
   02  CourseCode      PIC X(4).
   02  Gender          PIC X.

PROCEDURE DIVISION.
Begin.
    OPEN OUTPUT StudentFile
    DISPLAY "Enter student details using template below.  Enter no data
to end."

    PERFORM GetStudentDetails
    PERFORM UNTIL StudentDetails = SPACES
       WRITE StudentDetails
       PERFORM GetStudentDetails
    END-PERFORM
    CLOSE StudentFile
    STOP RUN.

GetStudentDetails.
    DISPLAY "Enter - StudId, Surname, Initials, YOB, MOB, DOB, Course,
Gender"
    DISPLAY "NNNNNNNSSSSSSSSIIYYYYMMDDCCCCG"
   ACCEPT  StudentDetails.
```

2.7 Zaman Paylaşımının (Timesharing) başlangıcı: BASIC

- BASIC (Beginner's All purpose Symbolic Instruction Code) dili ABD Dartmouth College'de iki matematikçi Kemeny ve Kurtz tarafından, bilim ile uğraşmayan kişilerin sayısal hesaplama gereksinimlerini karşılamak için geliştirilmiş bir programlama dilidir.
- **Kolay öğrenilmesi** ve **küçük bellekli bilgisayarlarda kullanılabilmesi** yaygın olarak kullanılmasını sağlamıştır.
- Tasarım amaçları:
  - Bilim dışı öğrencilerin öğrenmesi ve kullanması kolay olan bir dil
  - "güzel ve arkadaşça" olmalı
  - Ödev için hızlı çalıştırılabilir olması
  - Ücretsiz ve kişisel erişim
  - Kullanıcının zamanı bilgisayarın zamanından değerlidir
- Mevcut popüler diyalekt: Visual BASIC
- Süre paylaşımlı (time sharing) ilk yaygın kullanılan dil

BASIC

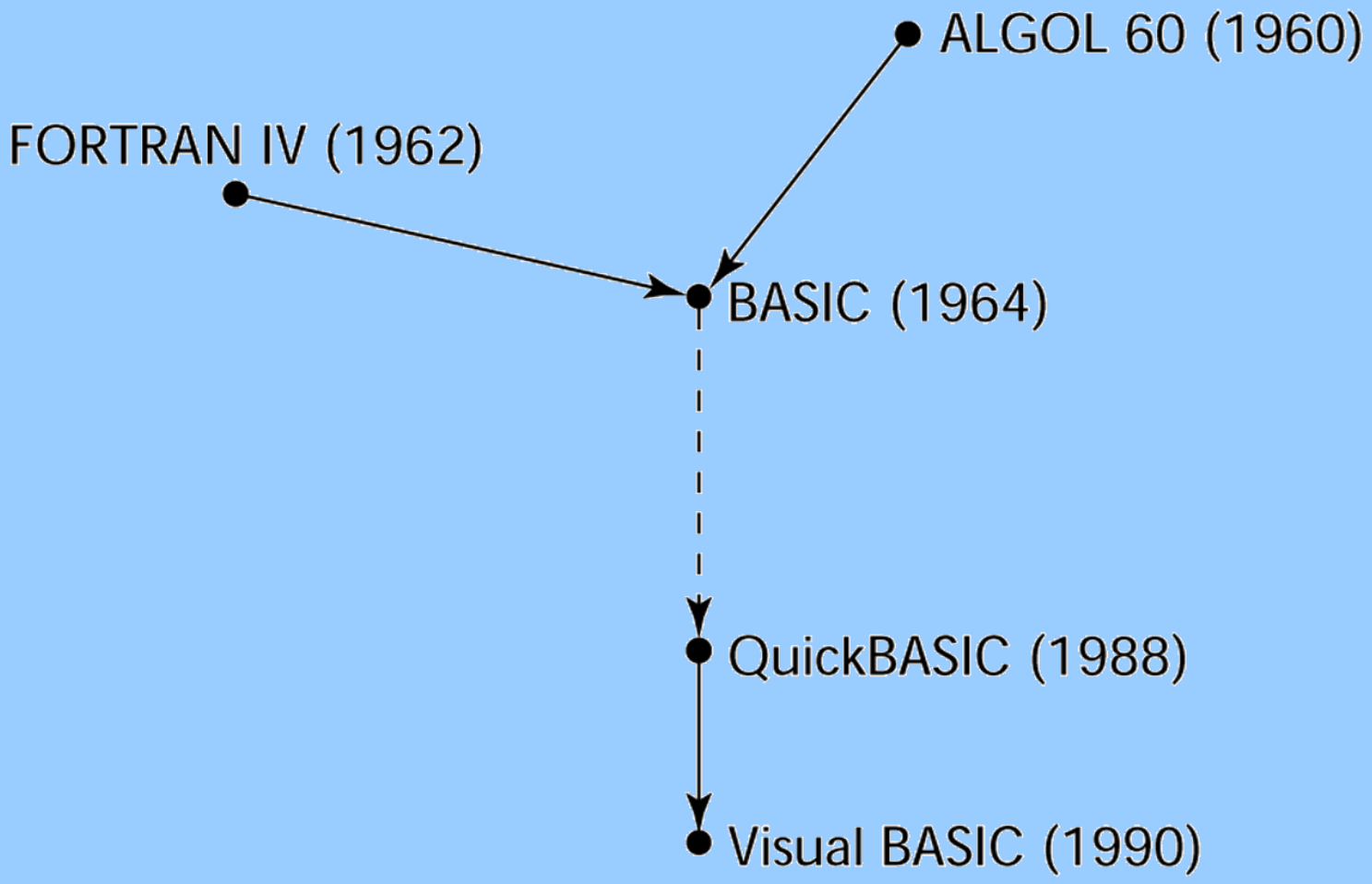
- Öğrencilerin bilgisayara daha kolay erişimlerini sağlamak ve basit ve etkin bir programlama dili ile program yazabilme isteklerine cevap vermek için tasarlanmış bir dildir.
- Sadece14 komuta (LET, PRINT, GOTO…) sahipti. Tek veri tipi (number= kayan noktalı ve tamsayı)
- BASIC, FORTRAN ve ALGOL'den bazı bileşenleri almıştır.

  FORTRAN'dan DO çevrimini, ALGOL'den ise "until" yerine "to"

BASIC

- Kolay bir dil ve genel maksatlı, belirli bir alana bağlı değil
- Uzman kişilere de hitap edebiliyor
- 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 pekçok versiyonları olmuştur. 1989'da ise nesne yönelimli uyarlama olan Visual BASIC ve 1998'de VB6.0 sunulmuştur.

BASIC'in Şeceresi

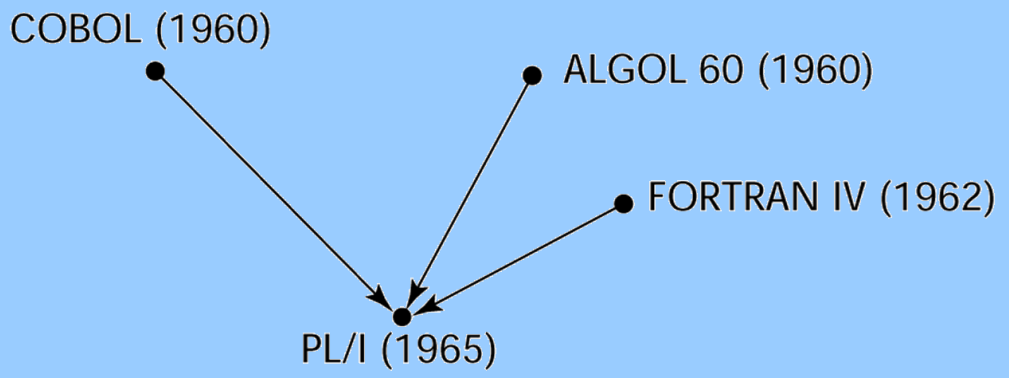


BASIC Örnek

```
REM BASIC Example program
   DIM intlist(99)
   result = 0
   sum = 0
   INPUT listlen
   IF listlen > 0 AND listlen < 100 THEN
      FOR counter =1 to listlen
          INPUT intlist(counter)
          sum = sum + intlist(counter)
      NEXT counter
      average = sum/listlen
      FOR counter =1 to listlen
          IF intlist(counter) > average
          THEN result = result +1
      NEXT counter
      PRINT "The number of values that are >
average is : "; result
  ELSE
      PRINT "ERROR - input list length is not
legal"
  END IF
END
```

2.8 Herkes için Herşey: PL/I

- Programming Language 1
- IBM ve SHARE tarafından tasarlandı. IBM'in yeni çıkardığı bilgisayarlar ile beraberinde gelen bir dildir.
- COBOL, Fortran IV ve ALGOL 60'tan sonra bilimsel ve işletme problemlerine çözüm sağlayabilmek için floating-point ve desima veritiplerini desteklemek üzere geliştirilmiş bir dildir.



PL/I

- 1964'te bilgisayar kullanma durumu (IBM'in bakış açısına göre)
  - Bilimsel kullanım
    - IBM 1620 ve 7090 bilgisayarları
    - FORTRAN
    - SHARE kullanıcı grubu
  - Ticari kullanım
    - IBM 1401, 7080 bilgisayarları
    - COBOL
    - GUIDE kullanıcı grubu

PL/I: Arkaplan

- 1963'le birlikte
  - Bilimsel kullanıcılar COBOL'deki gibi daha çok ayrıntılı I/O'ya; ticari kullanıcılar ise daha çok kayan nokta (floating point) ve diziye ihtiyaç duymaya başladılar
  - Öyle görünüyordu ki birçok mağaza iki çeşit bilgisayara, dile ve destek personeline ihtiyaç duymaya başlıyordu--çok pahalı
- Açıkça görülen çözüm
  - Uygulamaların iki çeşidini de yapabilecek yeni bir bilgisayar yapmak
  - Uygulamaların ikisini de yapabilecek yeni bir dil tasarlamak

PL/I: Tasarım İşlemi

- 3 X 3 Committee tarafından beş ayda tasarlandı
  - IBM'den üç üye, SHARE'den üç üye
- İlk kavram
  - Fortran IV'ün bir uzantısı
- Başlangıçta NPL (New Programming Language) adı verildi
- 1965' de adı PL/I olarak değiştirildi
- En önemli özelliği çeşitli uygulama alanları için kullanılabilmesidir. 70'li yıllarda sayısal ve iş uygulamalarında yaygın olarak kullanılmıştır.

PL/I

- İlk PL :
  - Eşzamanlı olarak çalışabilen altprogramlara izin verebilmektedir.
  - run-time hatalarından başka 23 farklı tip.
  - Rekürsif olarak kullanılabilecek prosedürleri desteklemektedir.
  - Pointer kullanımına izin vermektedir.
  - Bir matrisin bir satırına vektor olarak işaret mümkün
  - Hafıza gereksinimi yüzünden karmaşık olabilmektedir.
- Tasarım yönünden iyi değildir.

PL/I: Değerlendirmesi

- PL/I katkıları
  - İlk birim-düzeyli eş zamanlı olma (unit-level concurrency)
  - İlk istisna işleme (exception handling)
  - Anahtar-seçmeli (Switch-selectable) özyineleme (recursion)
  - İlk işaretçi (pointer) veri tipi
  - İlk çapraz dizi bölümleri (array cross sections)
- Zayıflıkları
  - Birçok yeni özellik zayıf tasarlanmıştı
  - Aşırı geniş ve aşırı karmaşıktı
    - PL/I dili karmaşık bir çok yapının eklenmesiyle zamanla karmaşık bir dil olmuştur.

2.9: İlk İki Dinamik Dil: APL ve SNOBOL

- Dinamik tip belirleme (dynamic typing) ve dinamik bellek ayrımı (storage allocation) ile karakterize edilir
- Değişkenlerin tipi yoktur
  - Değişkene bir değer atandığı zaman değişken tip edinir
- Bir değişkene değer atandığı zaman bellekte ona yer ayrılır

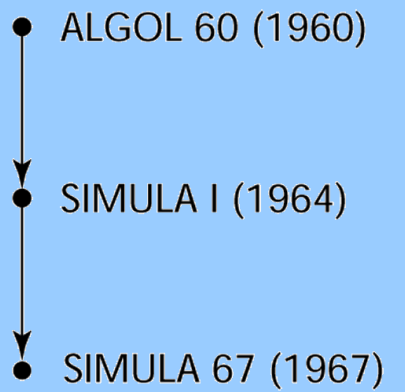
APL: **A** **P**rogramming **L**anguage (Bir programlama dili)

- 1960'larda IBM de Ken Iverson tarafından bir donanım tanımlama dili olarak tasarlanmıştır
  - Çok anlamlıdır (hem skaler (sayısal) hem de çeşitli boyutlarda diziler için birçok operatör)
  - Programların okunması çok zordur
- Sayısal işlemler özellikle de **dizi (matris) işlemleri** için güçlü olanaklar sağlayan bir dildir.
- Tek bir operatör ile matris transpozesi elde edilebilir.
- Dil ifadelerini oluşturan pek çok sayıda operatör bulunduran karmaşık söz dizimi dolayısıyla programların güç anlaşılması ve dinamik kaynak gereksinimi nedeniyle bilgisayar kaynaklarının etkin olarak kullanılamaması dezavantaj olmuştur.
- Halen kullanımdadır; çok küçük değişiklikler vardır

SNOBOL

- Bell Laboratuvarlarında Farber, Griswold, ve Polensky tarafından string ve text işleme dili olarak tasarlanmıştır
- String desen-eşleştirme (pattern matching) için güçlü operatörler
- Alternatif dillerden daha yavaştır (ve bu yüzden artık yazım editörleri tarafından kullanılmamaktadır)
- Hem APL hem SNOBOL dinamik tipleme ve dinamik bellek tahsisini desteklemektedir.
- Yorumlayıcı kullanan bir dildir
- Veri ve değişkenler üzerinde tip bakımından bir sınırlama bulunmuyordu
- Bu dilin devamı:awk, icon, perl,
- Halen bazı metin işleme işleri için kullanılır

2.10 Veri Soyutlama (Data Abstraction) nın başlangıçları: SIMULA 67



- Norveç'de Nygaard ve Dahl tarafından asıl olarak sistem simülasyonu için tasarlanmıştır
- ALGOL 60 (Blok yapısı ve kontrol ifadeleri buradan alınmıştır) ve SIMULA I 'e dayalıdır
- Birincil Katkıları
  - Eş yordam (Co-routines) – bir çeşit alt program
    - Daha önce durdurulduğu yerden itibaren yeniden çalışmaya başlayan altprogram
  - Sınıf (class) adı verilen bir yapı içerisinde geliştirilmiştir
  - Sınıflar veri soyutlamanın (data abstraction) temelleridir
  - Sınıflar hem lokal veri hem de fonksiyonellik içeren yapılardır
  - Nesneler, miraslar

Simula Örnek

```
Begin
     while 1 = 1 do
     begin
          outtext ("Hello World!");
          outimage;
     end;
End;
```

```
class Lab;
begin
   text Nam, Street, Town, County, Code;
   procedure ReadLabel;
   begin      Nam :- InLine;
                   Street :- InLine;
                   Town :- InLine;
                  County :- InLine;
                  Code :- InLine
   end++of++ReadLabel;
   procedure  WriteLabel;
   begin
       OutLine(Nam);
       OutLine(Street);
       OutLine(Town);
       OutLine(County);
       OutLine(Code)
    end++of++WriteLabel;
end--of--Lab;
```

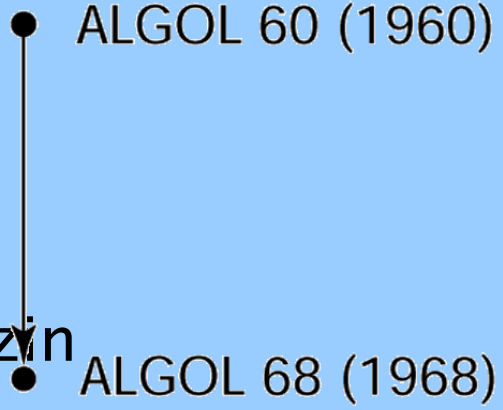
```
begin
   integer Int1;
   comment The first SIMULA program written for this book;
   Int1:=3;
   OutInt(Int1,4);
   OutImage
end
```

2.11 Ortogonal(Orthogonal) Dizayn: ALGOL 68

- ALGOL 60'ın devam eden gelişmesinden meydana gelmiştir fakat onun üstkümesi değildir
- Bazı yeni fikirlerin kaynağıdır (dilin kendisinin hiçbir zaman yaygın kullanıma ulaşamamasına rağmen)
- Tasarım ortogonallik (orthogonality) kavramına dayanır
  - Birkaç prensip kavram, birkaç birleştirici mekanizma

ALGOL 68 Değerlendirmesi

- **Katkılar**
  - kullanıcı-tanımlı veri yapılarını destekleyen ilk dil
  - Referans tipleri
  - Dinamik diziler (flex (esnek) arrays) e izin veren ilk dil
- **Yorumlar**
  - ALGOL 60 dan daha az kullanım
  - Öğrenilmesi oldukça zor olan bir gramer ve dil yapısına sahiptir.
  - Sadece bilimsel uygulamalar hedeflenerek tasarlanmış bir dil
  - Sonra gelen dillerde çok etkisi olmuştur, özellikle Pascal, C, ve Ada

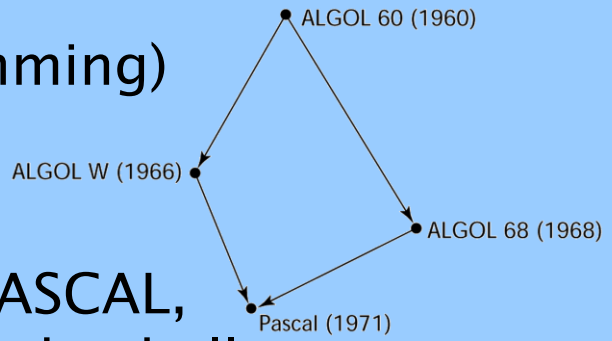


2.12 ALGOL'lerin ilk torunlarından bazıları

- ALGOL dilleri bütün zorunlu (imperative) dilleri etkiledi
  - Pascal
  - C
  - Modula/Modula 2
  - Ada
  - Oberon
  - C++/Java
  - Perl (bir yere kadar)

Pascal – 1971

Niklaus Wirth

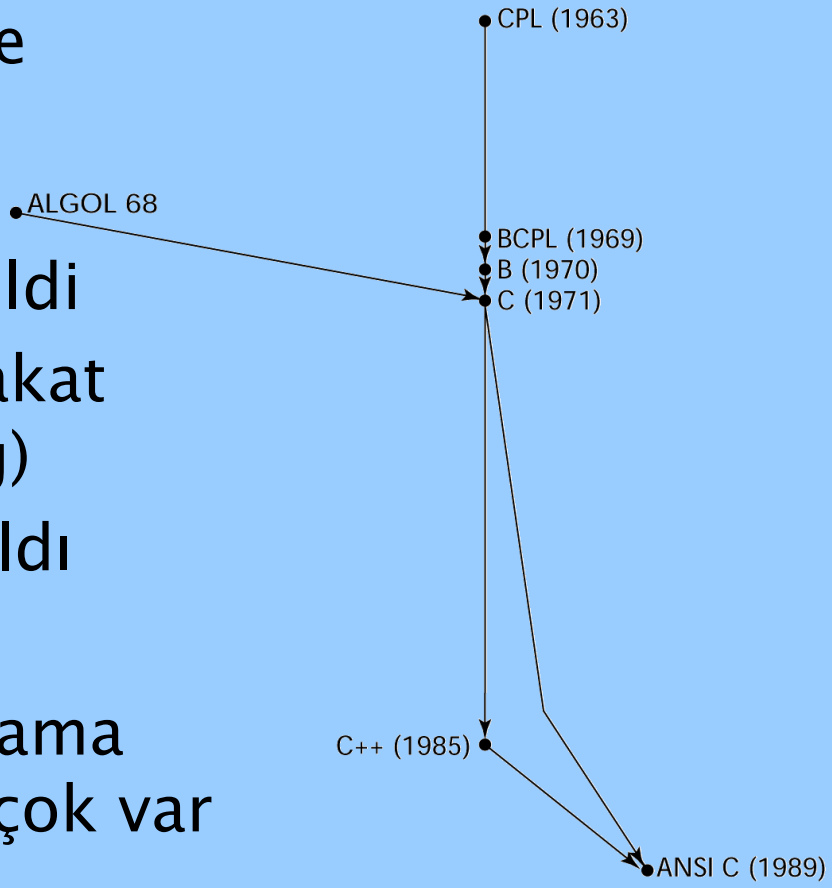


- ALGOL 68 dilinin tasarımında çalışan araştırmacı Nicholas Wirth tarafından öğretim amaçlı olarak tasarlanan PASCAL'ı geliştirildi
  - Yapısal programlama (structured programming) öğretmek için tasarlandı
- Küçük, basit, yenilik getirmeyen bir dil
- Önceki bir çok dilin özelliklerini barındıran PASCAL, COBOL ve PL/I'dan kayıt yapısını, ALGOL dilinden kullanıcı tanımlı tip kavramını ve case deyimini almıştır.
- Programlama öğretmede en çok etkisi oldu
  - 1970lerin ortalarından 1990 ların sonlarına kadar, programlama öğretmek için kullanılan en yaygın dildi
- Basit ve kolay anlaşılabilir bir dil olmasına rağmen bazı özelliklerinin olmamasından dolayı ticari uygulamalarda kullanımı sınırlı kalmıştır.

C – 1972

K. Thompson and D. Ritchie

- Sistem programlama için tasarlandı (Bell Laboratuvarlarında Dennis Richie tarafından)
- Temel olarak BCLP, B'den, aynı zamanda ALGOL 68'den geliştirildi
- Esnek yapı, güçlü operatörler, fakat zayıf tip kontrolü (type checking)
- Başlangıçta UNIX üzerinden yayıldı
- Birçok uygulama alanı
- PASCAL ve C dillerinin programlama literatürüne katkısı azdır. Daha çok var olan dillerin yapıları bir araya getirilmiştir.



Modula

- Yazılım Mühendisliği alanında önemli deneyimler sonucu ortaya çıkan ve güçlü bir tip ayrımı ve tip kontrolü mekanizmasına sahip bir dildir.
- Dinamik dizi kullanabilme (dinamik bellek yönetimi) ve eşzamanlılık özelliklerine sahiptir.
- Lilith adı verilen yeni bir bilgisayar sitemi için tasarlanmış fakat kullanılmamıştır.
- Modula 2 Pascal'a göre daha başarılı ve kullanıcıya esneklikler sunmakta. Modula 2'nin syntax'ı daha esnektir.
- Gerçek uygulama yazılım sistemleri için geliştirilmiştir.
- Soyut veri tipini destekliyor fakat miras alma (inheritance) özelliğini desteklemiyordu. Nesneye yönelim yok

Oberon

- Modula-2'ye dayanmakta fakat ondan daha basit idi.
- OOP'yı desteklemek için tipleme genişletilmişti.Tip ayrımı güçlüdür. OOP'i tam olarak destekler.
- Geleneksel veri türlerini ve kontrol yapılarının çoğunu içerir. Dinamik bellek yönetimi vardır.
- Endüstriyel gücü yüksek bir dil idi. Uygulama geliştirme amacıyla geliştirilmiş bir dildir.
- Veritabanı, ağ haberleşmesi, 3 boyutlu grafikler oluşturma ve nesne yönetimi için destek sağlayan zengin bir kütüphanesi vardır.
- Soyut veri türlerini destekler.

2.13 Mantık(Logic) temelli programlama: Prolog

- Comerauer ve Roussel (University of Aix-Marseille), Kowalski (University of Edinburgh) nin yardımıyla geliştirilmiştir
- Biçimsel mantığa (formal logic) dayalıdır
- Mantık yürütme ve ispatlama tekniklerini kullanır
- Prosedürel değildir
- Dil kuralar ve gerçeklerden olşur.
- PROLOG programı, bir nesneler kümesi ile bu nesnelerle ilişkili hedeflere nasıl erişilebileceğini tanımlayan kurallar kümesinden oluşur.
- Verilen sorguların (query) doğruluğunu anlamak için bir sonuç çıkarma kullanan akıllı bir veritabanı sistemi olarak özetlenebilir
- 1972 yılında tanıtılan PROLOG dilinin temel uygulama alanı, doğal dil işlemedir.
- Veritabanlarından uzman sistemlere kadar çeşitli uygulama alanlarında günümüze kadar kullanılmıştır.
- Çok verimsiz, dar uygulama alanları

Prolog Örnek

```
grandfather(X, Z) :- (mother(X, Y); father(X,Y)),
father(Y,Z).
```

```
mother(maria, ana).
father(juan, ana).
father(tom, juan).
```

grandfather(tom, ana)?

```
parent (X,Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
grandfather(X, Z) :- parent (X, Y),  father(Y,Z).
```

2.14 Tarihin en büyük tasarım çabası: Ada

- ABD Savunma Bakanlığı tarafından gömülü (embedded) sistemler için geliştirilmiştir. (1974~1979)
- Yüzlerce insan, çok para, ve yaklaşık sekiz yıl içeren muazzam tasarım çabası
  - Strawman gereksinimleri (Nisan 1975)
  - Woodman gereksinimleri (Ağustos 1975)
  - Tinman gereksinimleri (1976)
  - Ironman gereksinimleri (1977)
  - Steelman gereksinimleri (1978)
- İlk programcı olarak bilinen Augusta Ada Byron'dan (1815-1851)sonra Ada adı verildi.

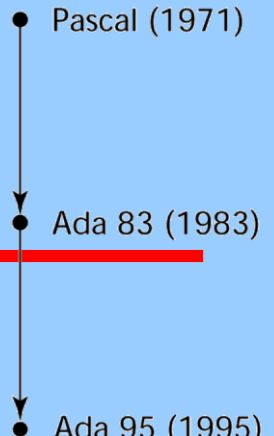
Ada Değerlendirmesi

- **Katkılar**
  - Paketler – veri soyutlama desteği
  - İstisna İşleme (Exception handling) – ayrıntılı
  - Soysal (Generic) program birimleri
    - Generic (Şablon, soysal) program birimleri sayesinde yazılımın yeniden kullanılabilmesini buluşma yeri (rendezvous) mekanizmasının ilavesiyle eşzamanlı çalışmayı desteklemektedir.
  - Eş zamanlılık (Concurrency) – görevleme (tasking) modeli ile
  - Blok yapılı
  - Büyük boyutlu yazılımlar için uygundur
- **Yorumlar**
  - Rekabetçi tasarım
  - Yazılım mühendisliği ve dil tasarımı hakkında sonradan bilinen her şeyi içeriyordu
  - İlk derleyiciler çok zordu; ilk gerçekten kullanılabilen derleyici dil tasarımının tamamlanmasından yaklaşık beş yıl sonra geldi

Ada Değerlendirmesi

- Çok geniş ve çok karmaşık bir dil. (Özellikle yazım kuralları)
- Çeşitli durumlara uygulanabilecek hazır şablonlara (template) sahiptir.
- İlk sıralar ADA derleyicileri kod üretmekte verimsiz idi.
- En çok gömülü sistemlerde başarılı olmuştur.
- Veri tipleri konusunda çok zengindir. Çok-iş işleme özelliğine sahiptir.

Ada 95



- Ada 95 (1988 de başladı)
  - Tip türetme (type derivation) üzerinden OOP desteği
  - Paylaşılan veri için daha iyi kontrol mekanizmaları
  - Yeni eş zamanlılık (concurrency) özellikleri
  - Daha esnek kütüphaneler
  - Altprogramların dinamik kapsam bağlama kurallarına göre çağrılması mekanizması da ADA 95'in özelliklerine katılmıştır
- Popülerliği azaldı çünkü DoD daha fazla kullanımına ihtiyaç duymadı buna karşın C++ 'ın popülaritesi arttı

Ada Örnek

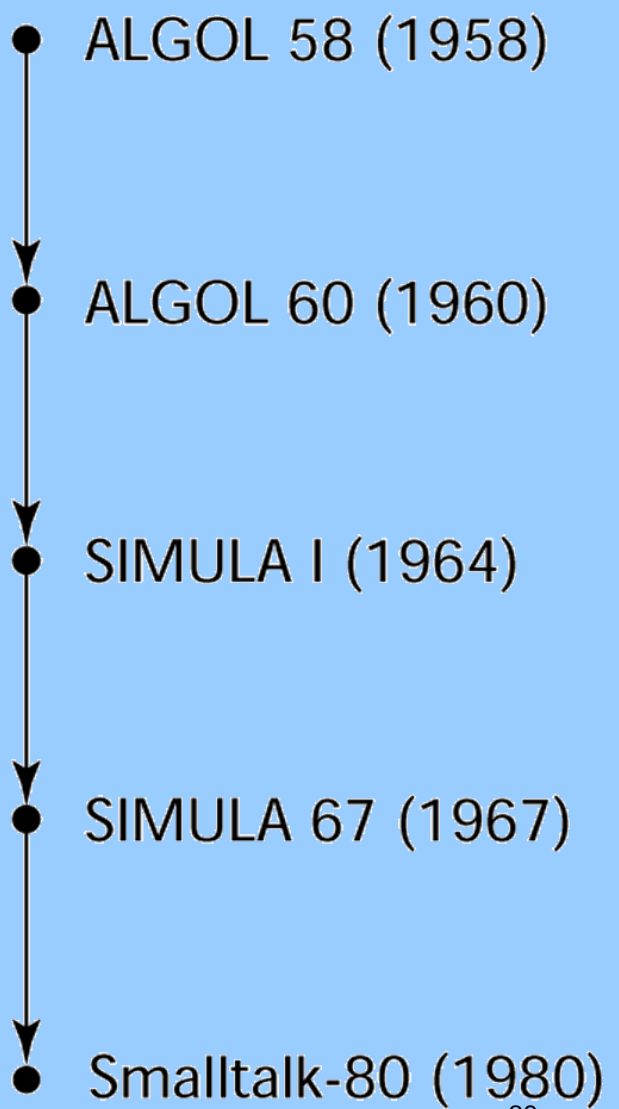
```
with Text_To;
use Text_To
    procedure hello is
    begin
       put(″Hello World″);
    end hello
```

```
package ArrayCalc is
        type Mydata is private;
        function sum return integer;
        procedure setval(arg:in integer);
        private
                size: constant:= 99;
                type myarray is array(1..size) of integer;
                type Mydata is record
                        val: myarray;
                        sz: integer := 0;
                        end record;
                v: Mydata;
end;
```

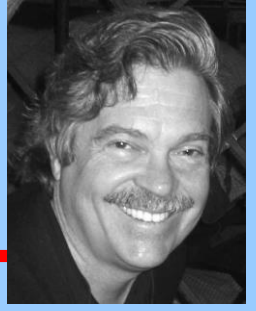
```
package body ArrayCalc is
        function sum return integer is
                temp: integer;
        -- Body of function sum
                begin
                temp := 0;
                for i in 1..v.sz loop
                        temp := temp + v.val(i);
                        end loop;
                        v.sz:=0;
                        return temp;
                        end sum;
                procedure setval(arg:in integer) is
                        begin
                        v.sz:= v.sz+1;
                        v.val(v.sz):=arg;
                        end setval; end;

with Text_IO; use Text_IO;
with ArrayCalc; use ArrayCalc;
procedure main is
        k, m: integer;
begin -- of main
        get(k);
        while k>0 loop
                for j in 1..k loop
                get(m); put(m,3);
                setval(m);
                end loop;
                new_line; put("SUM =");
                put(ArrayCalc.sum,4);
                new_line; get(k);
                end loop;
end;
```

2.15 Nesneye-dayalı Programlama: Smalltalk

- Xerox PARC'da, önce Alan Kay, sonra Adele Goldberg tarafından geliştirildi
- Bir nesneye-dayalı dilin ilk tamamen implementasyonu (veri soyutlama, miras (inheritance), ve dinamik tip bağlama (binding))
- Grafiksel kullanıcı arayüzü tasarımına öncülük etmiştir
- Sadece bir dil değil, aynı zamanda yazılım geliştirme aracıdır
- OOP 'yi yükseltmiştir



Smalltalk



Alan Kay

- İlk olarak Smalltalk-80 versiyonu Xerox dışında ticari amaçlı yazılımlarda kullanıldı. Halen yazılım geliştirimleri için yeni teknolojileri de içererek kullanılmaktadır.
- **Smalltalk tamamen nesne yönelimli olan ve ticari ilk programlama dilidir.**
- Geliştirme ortamı nesne yönelimli bir alt yapısı olması nedeni ile çok gelişmiştir (tarayıcı, editör, debugger, açık kaynak kod).
- **Kaynak kodunun açık olması iyi bir eğitim ortamı haline gelmesini sağlamıştır. Diğer dillere göre çok basit bir sentaksı vardır ve Java gibi Geniş ve sürekli genişleyen bir programlama kütüphanesine sahiptir. Platform bağımsız bir dildir.**
- Bu dilde nesne yönelimli programlamanın 3 temel karakteristiği olan veri soyutlama (data abstraction), kalıtım (inheritance) ve dinamik bağlama(dynamic binding) kavramlarının hepsi bulunmaktadır.

2.16 Zorunlu (Imperative) ve nesneye dayalı (Object-Oriented) özellikleri birleştirmek: C++

- Bell Labs'da Stroustrup tarafından in 1980 de geliştirilmiştir
- C ve SIMULA 67 den geliştirilmiştir
- Nesneye dayalı programlama olanakları, kısmen SIMULA 67'den alınmıştır
- İstisna yakalama (exception handling) sağlar
- Hem prosedürel (fonksiyona izin verir) hem de OO programlamayı desteklediği için geniş ve karmaşık bir dildir.
- Popülaritesi OOP ile birlikte hızla artmıştır
- ANSI standardı Kasım 1997'de onaylandı

C++

- Çoklu miras alma desteklenmekte
- Operatör ve metotların üstüste bindirilmesi desteklenmekte
- Sınıflar ve metotlat template edilebilir.
- Kuvvetli tip ayrımı, dinamik bellek yönetimi, hazır şablonlara sahip olma ve çok biçimlilik (polymorphism) özellikleri vardır.
- C'de bulunan özelliklerin çoğu burada vardır.
- PL/1 gibi geniş ve kompleks.
- Ada ve Java'dan daha az güvenli
- Microsoft'un sürümü (.NET le 2002 de çıkan): Yönetilmiş (Managed) C++
  - Delegeler (delegates), arayüzler (interfaces), çoklu miras (multiple inheritance) yoktur

Eiffel (Bertrand Meyer – 1992 de)

- Direk olarak başka bir dilden geliştirilmemiştir
- Imperative ve OO özellikleri birleştiren Hybrid PL
- C++'tan küçük ve basittir, ama halen daha güçlüdür
- C++ kadar popüler değildir çünkü birçok C++ hayranı önceden C programcılarıydı
- Soyut veri yapılarını, kalıtımı ve dinamik bildirimleri destekler (OO)
- Altprogram ve çağırıcı (caller) arasındaki iletişim için bildirimler (assertions) kullanır.
- C++'dan daha küçük ve basittir. Fakat ifade edilebilirliği ve yazılabilirliği neredeyse eşittir.

Delphi (Borland)

- Pascal artı OOP yi destekleyen özellikler
- C++'tan daha zarif, daha az kompleks ve güvenlidir
- Emir esaslı ve OO PL'nin başarılı bir biçimde birleştirilmesidir.
- Pascal'dan türemiştir. Bu yüzden, dizi elemanlarının kontrolünde, pointer aritmetiği ve tip zorlamalarında C ve C++'tan daha emniyetlidir.
- Kullanıcı tanımlı operatörlere, generic altprogramlara ve parametize edilmiş sınıflara izin vermez.
- Daha iyi ve daha kolay yazılım geliştirtme için bir Graphical User Interface (GUI) sağlamaktadır.

2.17 Bir Zorunlu nesneye-dayalı dil (Imperative-Based Object-Oriented): Java

- 1990'ların başında Sun'da geliştirildi
  - C ve C++ gömülü electronik aygıtlar için yeterince memnun edici değildi
- C++ temellidir
  - Önemli derecede basitleştirilmiştir (**`struct`**, **`union`**, **`enum`**, işaretçi (pointer) aritmetiği, ve C++'ın atama zorlamalarının yarısını içermez)
  - *Sadece* OOP yi destekler
  - Referansları vardır, işaretçiler (pointers) yoktur
  - Appletler ve bir eş zamanlılık (concurrency) formu için destek içerir

Java Değerlendirmesi

- C++'ın güvensiz özelliklerini elemiştir
- Eş zamanlılık (Concurrency) özellikleri
- Appletler için kitaplıklar, GUIler, veritabanı erişimi
- Taşınabilir: Java Sanal (Virtual) Makinesi kavranı, JIT derleyiciler
- WWW sayfaları için yaygın kullanılmaktadır
- Diğer alanlarda kullanımı başka dillere oranla daha hızlı artmaktadır

Java

- Daha küçük, daha basit ve daha güvenilir bir PL tasarlamak üzere C++ tabanlı olarak geliştirilmiştir.
- 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.
- Hem referans değişkenleri ile hem de ilkel tiplerle erişilebilen sınıflara sahiptir.
- C++'ta bulunan çoklu miras alma, operatörlerin üst üste bindirilmesi, ve makro önişlemcisi 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.
- Records, union or enumeration tipler yoktur.
- Prosedürel programlamayı desteklemez.
- Sadece tekli miras almayı destekler fakat çok gelişmiş bir GUI'ya sahiptir.
- İplik (**threds**) yapısına (bir pencere içerisinde yeni pencereler açılmasına izin var) sahip olduğundan eşzamanlılığı yönetmek kolaydır.
- **Çöp toplama (Garbage collection)** nesneler için belleği en iyi kullanımı sağlar.
- Tip dönüşümü kuvvetlidir.

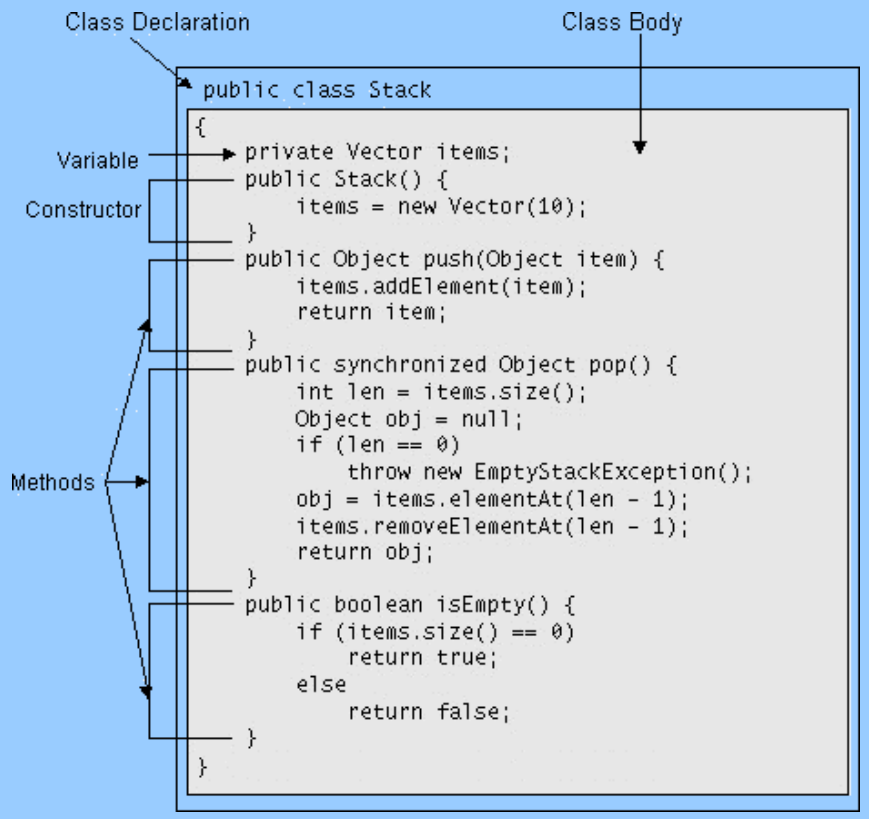
Java

- Java tipik olarak platformdan bağımsız olarak byte kodları biçiminde derlenir. Daha sonra bu byte kodlar bir Java görüntü makinesi adı verilen bir Java yorumlayıcısı tarafından kullanılacağı platformun makine dilindeki koduna çevrilir. Derlenmiş Java sınıflarının taşınabilirliğini garantileyen bir özellik vardır. Oda .class format adı verilen Java byte-kod dosya formatının kesin olarak tanımlanmış olmasıdır.
- Java "applet" adı verilen başka sistemler içinde gömülü programların geliştirilmesi için de kullanılmaktadır. Bu appletler Internet Explorer yada Netscape gibi web tarayıcı programlar içerisinde kullanılabilir.
- Java internet ve web programcılığında yaygın olarak kullanılmaktadır.

Java Örnek

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); //Display the string.
    }
}
```

```
import java.applet.Applet;
import java.awt.Graphics;
public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 50, 25);
    }
}
```



2.18 Ağ (Web) için Betik Diller (Scripting Languages):

- **Perl**
  - ALGOL ile sadece C üzerinden ilişkilidir
  - Bir yazı (scripting) dilidir
    - Bir yazı dosyası (*script* file) çalıştırılacak komutları içerir
    - Diğer örnekler: sh, awk, tcl/tk
  - Larry Wall tarafından geliştirilmiştir
  - Perl değişkenleri statik tiplidir ve örtülü (implicitly) tanımlanmıştır
    - Üç farklı isim alanı (namespace), bir değişkenin adının ilk karakteriyle gösterilir
  - Güçlü fakat tehlikeli
  - Genel amaçlı bir dil olarak yaygın kullanılmaktadır
  - Web CGI programlama için yaygın kullanımı kazanmış
  - Ayrıca UNIX sistem yönetimi dil için yedek olarak kullanıldı

2.18 Ağ (Web) için Betik Diller (Scripting Languages):

- **JavaScript**
  - Netscape ve Sun Microsystems ortaklığı
  - Web programlamada (istemci tarafı-client side) dinamik HTML dökümanları oluşturmak için kullanılır
  - Tamamen yorumlayıcıdır
  - Java ile sadece benzer sentaksı nedeniyle ilgilidir
  - Netscape ve Sun Microsystems 'in ortak çalışmaları sonucu 1995'de üretilmiştir. Orijinal adı "LiveScript"tir.
  - Dil bileşenlerinin içinde, web sayfalarının çeşitli kısımlarını ve özellikle HTML'i işleyecek ve kontrol edecek çeşitli olanaklar vardır.
  - Javanın veri tipleri bakımından sert kısıtlamalarına karşılık, (Örn.Bütün değişken tipleri derleme zamanında belirlenir), Javacript bu açıdan toleranslıdır.(dynamik olarak tipleme)
  - Java'nın HTML ile çalışma desteği zayıf iken, Javacript'in bu konudaki desteği tamdır.
  - Javascript nesneye yönelik ve blok-yapısal özellikte bir dildir.

2.18 Ağ (Web) için Betik Diller (Scripting Languages):

- **PHP (Personal Home Page)**
  - PHP: Hypertext Preprocessor
  - Rasmus Lerdorf tarafından 1994'te tasarlandı
  - Web uygulamaları (sunucu tarafı-server side) için kullanılır;
    - Genellikle Web üzerinden form işleme ve veritabanı erişimi için kullanılan bir sunucu tarafı HTML içine gömülü bir betik dilidir
    - Çıktı olarak HTML kodu üretir
  - Bir HTML dökümanı, okuyucu, hiper linkler boyunca diğer HTML dökümanlarına götürecek biçimde programlanabilir.
  - Gömülü bir HTML belgesi browser tarafından istendiğinde PHP kodu Web server üzerinde yorumlanır.
  - PHP'nin dizi yapısı JavaScript ve Perl dizi yapılarının bir birleşimidir.
  - PHP ile form erişimi oldukça kolaydır.
  - Tamamen yorumlayıcıdır

2.18 Ağ (Web) için Betik Diller (Scripting Languages):

- **Python**
  - Nesne tabanlı yorumlayıcıya sahip bir metin dilidir
  - Tip kontrol edilir ama dinamik yazılır
  - CGI programlama ve form işleme için kullanılır
  - Dinamik yazılabilir, ancak tipi kontrol edilir
  - Listeleri, değişkenler gurubu ve karmaları destekler
- **Lua**
  - Nesne tabanlı yorumlayıcıya sahip bir metin dilidir
  - Tip kontrol edilir ama dinamik yazılır
  - CGI programlama ve form işleme için kullanılır
  - Dinamik yazılabilir, ancak tipi kontrol edilir
  - Listeleri, değişkenler gurubu ve karmaları destekler, bütün bunları onun tek veri yapısı ve tabloları üzerinden yapar
  - Kolayca genişletilebilir

2.18 Ağ (Web) için Betik Diller (Scripting Languages):

- **Ruby**
  - Yukihiro Matsumoto (a.k.a, “Matz”) tarafından Japonya’da tasarlandı
  - Perl ve Python için yedek bir dil olarak başladı
  - Bir saf nesne yönelimli bir (Script) dil
    - Tüm veriler nesnedir
  - Birçok operatör kullanıcı kodu tarafından yeniden tanımlanabilen metotlar olarak uygulanır
  - Sade yorumlayıcıdır

2.19 Yeni milenyum için C-temelli bir dil: C#

- .NET geliştirme platformunun bir parçasıdır
- C++ , Java, ve Delphi temellidir
- Bileşen-temelli (component-based) yazılım geliştirme için dil sağlar
- Bütün .NET dilleri (C#, Visual BASIC.NET, Managed C++, J#.NET, ve Jscript.NET) Ortak Tip Sistemi(Common Type System (CTS)) kullanır, bu ortak bir sınıf kütüphanesi sağlar
- Yaygın kullanıma ulaşmıştır.

C#

- C ve C++ dil ailesinin ilk bileşen yönelimli (Component-oriented) dilidir.
- C ve C++'dan derlenmiş, basit, modern, nesne yönelimli ve tür güvenli bir programlama dilidir.
- Yüksek başarımlı Common Language Runtime (CLR); bir yürütme motoru, bir çöp toplayıcı (garbage collection), anında derleme, bir güvenlik sistemi ve zengin bir sınıf çerçevesi (.NET Framework) içerir. CLR temelden, birden çok dil desteğine kadar herşey için tasarlanmıştır.
- CLR'ı hedef alan diller: Visual C#, Visual BASIC .NET, Managed C++, J#.NET ve Jscript.NET
- Common Language Specification, CLS, dil işlevselliğinin yaygın bir düzeyini tanımlar. .NET Framework işlevselliğine tam erişim ve ve diğer uyumlu dillerle zengin birlikte çalışabilirlik vardır

C#

- C# otomatik bellek yönetimini kullanır. C# tür sistemi işaretçi türleri ve nesne adreslerinin doğrudan değiştirilmesine de izin verilir.
- C# tür sistemi bileşiktir. Her şey bir nesnedir. Kutulama ve kutuyu açma gibi kavramların yenilikçi kullanımı ile C#, her veri parçasının bir nesne olarak değerlendirilmesine olanak sağlayarak, değer türleri (value type) ve başvuru türleri (reference type) arasındaki açığı kapatır.

2.20 İşaretleme(Markup)/Programlama Hibrit Diller

- XSLT
  - eXtensible Markup Language (XML) (genişletilebilir işaretleme dili): bir metamarkup dili
  - eXtensible Stylesheet Language Transformation (XSTL)(genişletilebilir stilsayfası dil dönüşümü) XML dökümanlarını görüntülenebilmesi için dönüştürür
  - Programlama yapıları (örn., döngüler)
- JSP
  - Java Server Pages (Java Sunucu Sayfaları): dinamik web dökümanlarını destekleyen teknolojiler koleksiyonu
  - servlet: bir Web servera ait bir Java programı; servlet'in çıktısı browserda görüntülenir


COBOL
ALGOL
PL/1
ADA
DOD
FORTRAN
PASCAL
MODULA 2
C
UNIX
N.W.
LISP
APL
BASIC
FORTH
A⇒B
PROLOG
MACHINE LANGUAGE
PILOT
ASSEMBLE
LOGO
'85

Özet

- Geliştirme (development), geliştirme platformu (development environment), ve bazı önemli programlama dillerinin değerlendirilmesi
- Dil tasarımındaki mevcut sorunlara bakış açısı

Kaynaklar

- Roberto Sebesta, Concepts Of Programming Languages, International 10th Edition 2013
- David Watt, Programming Language Design Concepts, 2004
- Michael Scott, Programming Languages Pragmatics, Third Edition, 2009
- Zeynep Orhan, Programlama Dilleri Ders Notları
- Mustafa Şahin, Programlama Dilleri Ders Notları
- Ahmet Yesevi Üniversitesi, Uzaktan Eğitim Notları
- Erkan Tanyıldızı, Programlama Dilleri Ders Notları