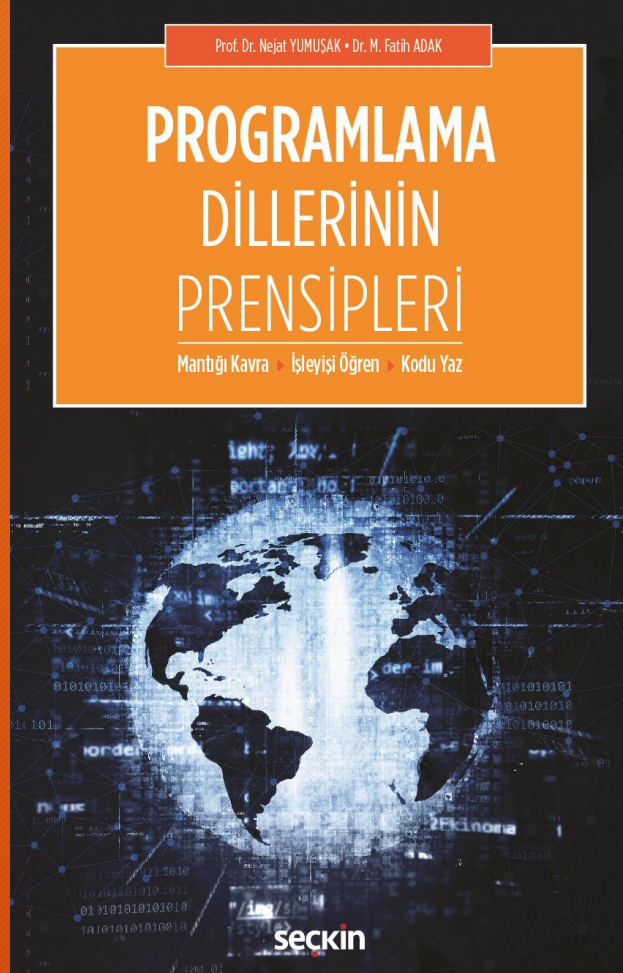


Programlama Dillerinin Prensipleri

Hafta 1 - Giriş

Dr. Öğr. Üyesi M. Fatih ADAK

Ders Kitabı



<https://www.seckin.com.tr/kitap/246539963>

Ders Akışı

Hafta	Konular
1	Giriş ve Programlama Kavramlarının Tanıtılması
2	Dillerin tarihçesi ve evrimi
3	Dil tanımlanması ve Dil çevrimi
4	Temel programlama kavramları, Veri tipleri ve yapıları
5	Bağlama Kavramları ve İsim Kapsamları
6	Yapısal programlama
7	Altprogramlar ve Modülasyon
8	Parametre aktarım yöntemleri
9	Nesne yönelimli programlama kavramları
10	Programlama dillerinde hata yakalama
11	Programlama dillerinde eşzamanlılık
12	Yorumlamalı Diller ve Python
13	Fonksiyonel programlama kavramları
14	Mantıksal programlama kavramları

Değerlendirme Sistemi

- 1. Ödev (Java) : % 20 (yaklaşık)
- 2. Ödev (C) : % 20 (yaklaşık)
- Kısa Sınav (Test) : % 5 (yaklaşık)
- Ara Sınav (Klasik) : % 15
- Final (Klasik) : % 40

- Ödevler bireyseldir.

Ödev ve Kod Derslerinde Araçlar

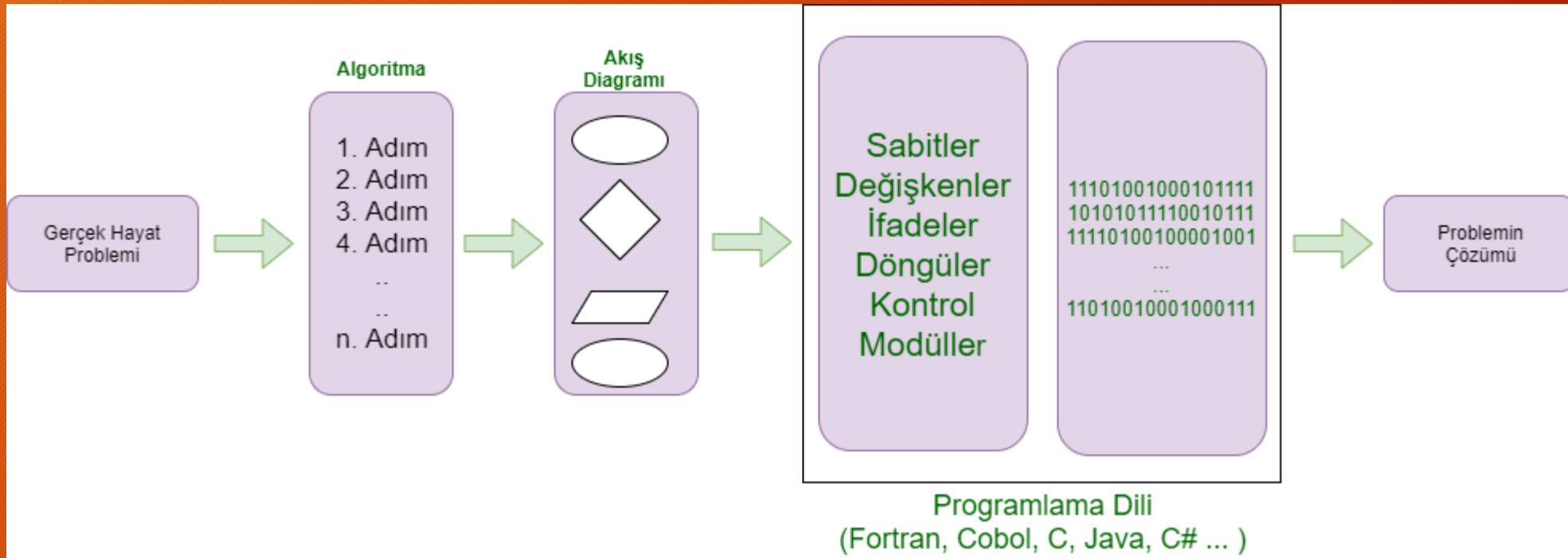
- Java için Eclipse 2020-12
 - <https://www.eclipse.org/downloads/>
- C Dili için
 - MinGW
 - <http://www.mingw.org/>
- Lisp Dili için
 - GNU Common Lisp 2.6.12
 - <https://www.gnu.org/software/gcl/>

Bu Haftaki İçerik

- Programlama Dillerinin Sınıflandırılması
- Programlama Dili Paradigmaları
- Programlama Dillerinin Değerlendirme Ölçütleri
- C Dili
- Java Dili
- Lisp Dili
- Prolog

Programlama Dili Nedir?

- Bir problemin çözümünün bilgisayardaki gerçekleştirimini ifade etmek üzere program oluşturmak için kullanılan araca denir.



Yazılımın Uygulama Alanlarına Göre Gruplandırılması

- Bilimsel ve mühendislik yazılımları
- Mesleki yazılımlar
- Yapay zeka yazılımları
- Görüntüsel yazılımlar
- Sistem yazılımları

Programlama Dilinin Amaçları

- Bir programlama dili makinalara talimat vermek için gerekli bir araçtır.
- Programcılar arasında iletişim için gerekli bir vasıtaadır.
- Yüksek seviyeli tasarımları ifade etmek için gerekli bir araçtır.
- Algoritmaları göstermeye yarayan bir notasyondur.
- Genel kavramlar arasındaki yakınlıkları ifade etmeye yarayan bir yoldur.
- Çözümlerin ve çözüm yollarının test edilmesi için gerekli bir araçtır.
- Bilgisayarlı cihazları kontrol etmek için gerekli bir vasıtaadır.

Programlama Dillerinin Sınıflandırılması

- Seviyelerine Göre Sınıflandırma
- Uygulama Alanlarına Göre Sınıflandırma
- Tasarım Paradigmalarına Göre Sınıflandırma

Programlama Dillerinin Seviyelerine Göre Sınıflandırma

- Makine Dili
- Alçak Seviyeli Diller
- Orta Seviyeli Diller
- Yüksek Seviyeli Diller
- Çok Yüksek Seviyeli Diller

Makine Dili

- Bir bilgisayarın doğrudan anlayabildiği bir dildir.
- Bilgisayarın ana dili olarak kabul edilir.
- Makine dili taşınabilir değildir ve makineye özgü yazılması gerekir.
- Makine dilinde kod yazmak çok zahmetli, çok zaman alıcı ve uğraştırıcıdır.

Alçak Seviyeli Programlama Dili

- Sembolik Dil (Assembly)

- Makine dili kullanımının getirdiği problemleri ortadan kaldırmak üzere yapılan çalışmalarda
- Önce makine dilinin anlaşılma zorluğunu kısmen de olsa ortadan kaldırmak üzere sembolik dil geliştirilmiştir.
- Sembolik dilde 0 ve 1'ler yerine İngilizce ifadeler yer almaktaydı.
- Burada bellekten okuma yazma yerine çok daha hızlı olması açısından register'lar kullanılır.

Sembolik dilde ekrana Merhaba yazdırılması
--

<pre>mesaj db 'Merhaba', 0x0d, 0x0a, '\$' mov dx, mesaj mov ah, 9 int 0x21</pre>
--

Orta Seviyeli Diller

- Sembolik diller bilgisayar kullanımını hızla arttırmıştır.
- Ancak çok basit işlemler için bile birçok komut gerekmektedir.
- Ayrıca sembolik diller her seferinde makine diline çevrilip öyle çalıştırılıyordu. Bu işlem program hızını 30 kat yavaşlatıyordu.
- Grace Hopper, bu problemin çözümü için derleyici fikrini ortaya attı.
- Program kodu bir kez derlenip makine diline çevrilecek ve bir daha bu işleme gerek kalmayacaktı.

Orta Seviyeli Diller

- Ada, C gibi diller örnek verilebilir.
- Daha az kayıpla makine diline çevrilebildiğinden daha hızlı çalışır.
- Program yazmak yine zordur fakat sembolik dile göre oldukça kolaydır.

Yüksek Seviyeli Diller

- Fortran ilk yüksek seviyeli dildir.
- Program yazmak daha kolay fakat orta seviyeli dillere göre program hızı daha yavaştır.
- Bu seviyedekiler 3. kuşak diller olarak kabul edilir.

Çok Yüksek Seviyeli Diller

- Genellikle algoritmik yapı içermeyen görsel bir ortamda yazılan dillerdir.
- 4. Kuşak olarak isimlendirilirler.
- Java, C#, Visual Basic, Access, Oracle Forms bu seviyeye örnek verilebilir.
- Program hızları makine dillerine göre oldukça yavaştır.

Programlama Dillerini Uygulama Alanlarına Göre Sınıflandırma

- Bilimsel ve Mühendislik Uygulama Dilleri
 - Pascal, C, Fortran
- Veritabanı Dilleri
 - MSSQL, Oracle Forms, XBASE
- Genel Amaçlı Programlama Dilleri
 - Pascal, C, Basic, Java
- Yapay Zeka Dilleri
 - Prolog, Lisp
- Modelleme Yapmak Üzere Geliştirilen Simülasyon Dilleri
 - GPSS, Simula67

Programlama Dillerini Uygulama Alanlarına Göre Sınıflandırma

- Makro Diller (Script Diller)
 - awk, Perl, Python, Tcl, Javascript
- Sistem Programlama Dilleri
 - C (UNIX işletim sisteminin %80'i C dili ile geri kalanı sembolik dil ile yazılmıştır.)
- Ticari Uygulamalara Yönelik Programlama Dilleri
 - Cobol

Programlama Dillerini Tasarım Paradigmalarına Göre Sınıflandırma

- Emir Esaslı Programlama
- Nesneye Yönelik Programlama
- Fonksiyonel Programlama
- Mantık Esaslı Programlama

Programlama Dillerini Tasarım Paradigmalarına Göre Sınıflandırma

- Emir Esaslı (Impretive) Programlama

Deyim 1
Deyim 2
.
.
.
Deyim n

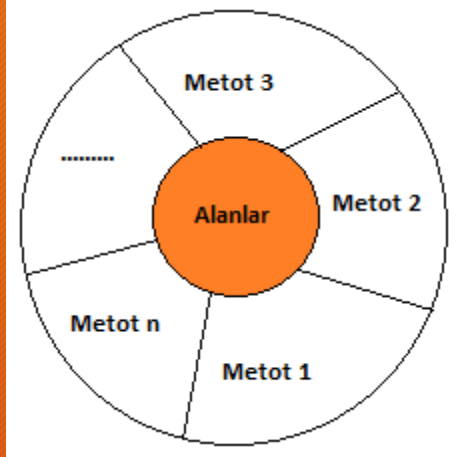
- Emir esaslı programlama dilleri işlem tabanlı olup, bir program bir dizi işlem olarak görülür.
- Bu diller yaygın olarak kullanılan ilk dil grubudur.
- C, Fortran, Pascal, Cobol örnek olarak verilebilir.

- Örneğin, atama işlemi bir deyimdir.

Farklı dillerde atama işlemleri
C dili x=13
Pascal dili x:=13
APL dili x<--13
Scheme veya Lisp dili (setq x 13)

Programlama Dillerini Tasarım Paradigmalarına Göre Sınıflandırma

- Nesneye Yönelik Programlama



- Temeli simula67 programlama dilidir.
- Nesnelerin sınıfı ve alt sınıflara gruplanması, nesneye yönelik programlamanın temel noktasıdır.
- Karmaşık veri nesneleri ve bu veriler üzerinde çalışacak işlemler (metotlar) tasarlanır.



Nesneye yönelik programlamanın genel yapısı

Programlama Dillerini Tasarım Paradigmalarına Göre Sınıflandırma

- Fonksiyonel Programlama

- Veriler ve sonucu elde etmek için veriye uygulanacak fonksiyonel dönüşümler bu paradigmanın temelini oluşturur.
- Lisp, Scheme ve ML dilleri bu paradigmaya örnektir.



- Fonksiyonel programlamanın temelini oluşturan parçalar



Programlama Dillerini Tasarım Paradigmalarına Göre Sınıflandırma

- Mantık Esaslı Programlama
 - Bir işin nasıl yapılacağıının belirtilmesi yerine, ne yapılması istendiğinin belirtilmesi olarak görülür.
 - Belirli bir koşulun varlığını kontrol ederek ve koşul sağlanıyorsa, uygun bir işlem gerçekleştirerek çalışırlar.
 - Emir esaslı programlamaya benzer fakat deyimler sıralı olarak işlenmez.

Mantıksal paradigmayı destekleyen dillerin sözdizimi
Şart_1 → Hareket_1
Şart_2 → Hareket_2
Şart_3 → Hareket_3
...
...
...
Şart_n → Hareket_n

Programlama Dillerinin Değerlendirme Ölçütleri

- Her programlama dili bir düşünce biçimi olduğundan binlerce programlama dili vardır denilebilir.
- Çözülecek problemin tipine ve uygulama alanına göre programlama dilleri arasında seçim yapmak için çeşitli değerlendirme ölçütlerine ihtiyaç duyulmaktadır.
- Her alan için en iyi olan bir programlama dili yoktur.

Programlama Dillerinin Değerlendirme Ölçütleri

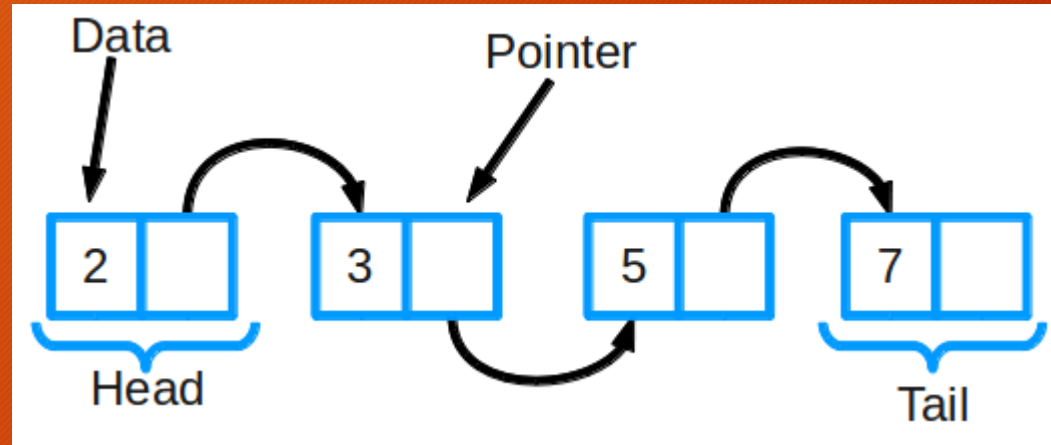
- İfade Gücü (Expression Power)
- Veri Türleri ve Yapıları (Data Types and Structures)
- Giriş/Çıkış Kolaylığı (Input/Output Facilities)
- Taşınabilirlik (Portability)
- Alt programlama Yeteneği (Modularity)
- Verimlilik (Efficiency)
- Okunabilirlik (Readability), Yazılabilirlik
- Esneklik (Flexibility)
- Öğrenme Kolaylığı (Pedagogy)
- Genel Amaçlılık (Generality)
- Yapısallık (Structurulness)
- Nesne yönelimlilik (Object Orientation)

Programlama Dillerinin Değerlendirme Ölçütleri

- İfade Gücü (Expression Power)
 - Algoritmayı tasarlayan kişinin niyetlerini açık bir biçimde yansıtabilmesine olanak tanınması
 - Günümüz popüler programlama dillerinin ifade gücü yüksektir.

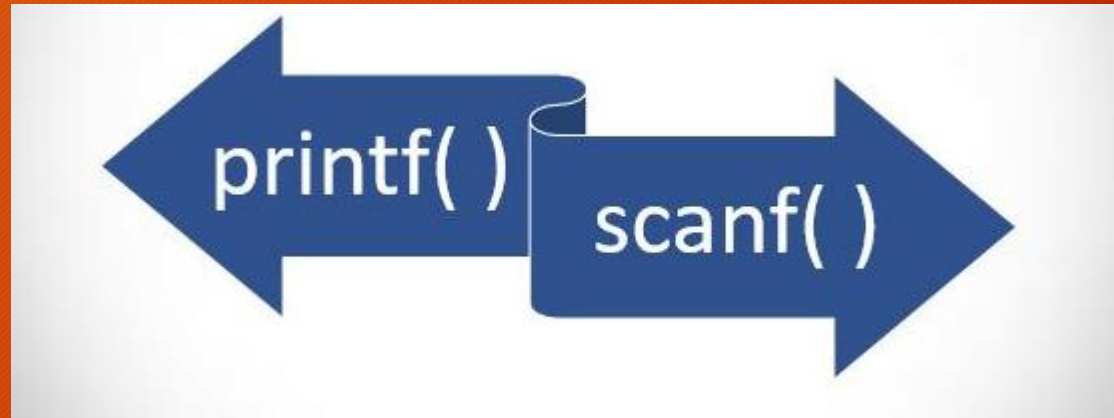
Programlama Dillerinin Değerlendirme Ölçütleri

- Veri Türleri ve Yapıları (Data Types and Structures)
 - Çeşitli veri türlerini (tamsayı, gerçek sayı, karakter...) ve veri yapılarını (diziler, bağlı liste, kuyruk, yapılar vs.) destekleme yeteneğidir.



Programlama Dillerinin Değerlendirme Ölçütleri

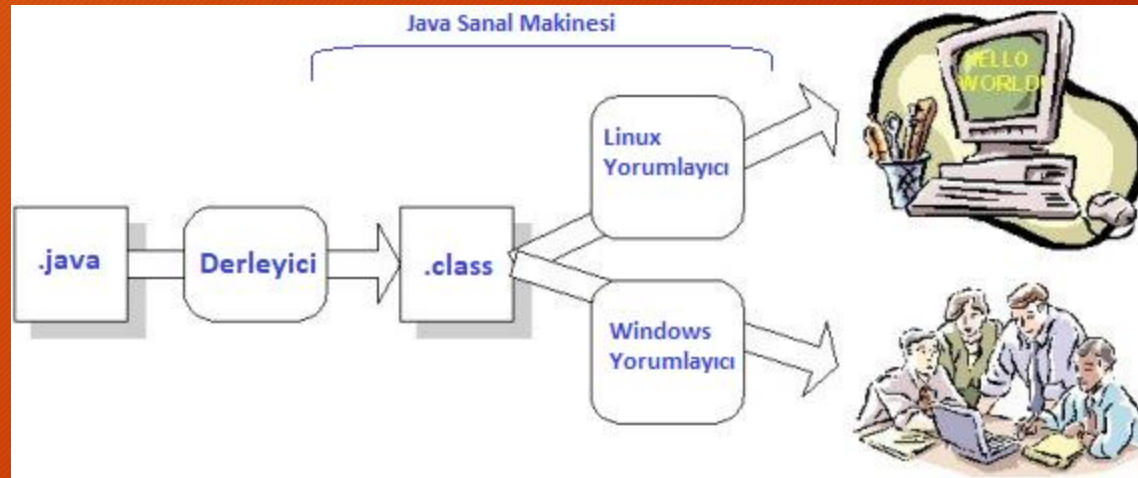
- Input Output Kolaylığı
 - Program yazmayı kolaylaştıran ve ifade gücünü arttıran bir özelliktir
 - Örneğin C dilinde bu ölçüt çok yüksek değildir.



Programlama Dillerinin Değerlendirme Ölçütleri

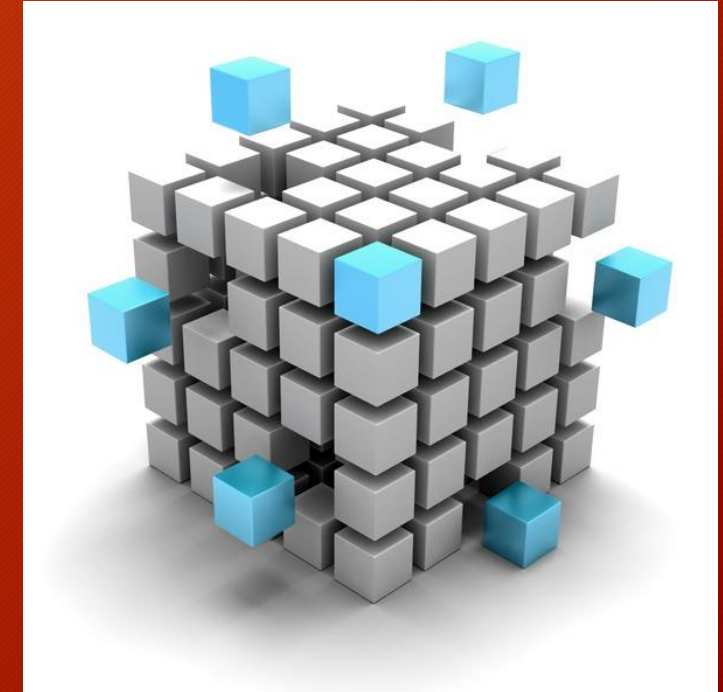
- Taşınabilirlik

- Bu terim kaynak kod için kullanılır.
- Bir sistemde yazılmış kaynak kodun bir başka sistemde de sorunsuz derlenip çalışmasıdır.
- Dillerin seviyesi düştükçe taşınabilirlik azalır.
- Hiçbir dil için mükemmel taşınabilirlik mümkün değildir.



Programlama Dillerinin Değerlendirme Ölçütleri

- Alt Programlama Yeteneği
 - Bir programı parçalar halinde yazmayı desteklemesidir.
 - Yapısal Programlama tekniğinin vazgeçilmez bir parçasıdır.
 - Yazılacak kodu oldukça azaltır.
 - Program kodlarının anlaşılmasını kolaylaştırır.



Programlama Dillerinin Değerlendirme Ölçütleri

- Verimlilik
 - Amaç koda dönüştürülmüş programların hızlı çalışabilmesine verimlilik denir.
 - Verimlilik derleyici, dil seviyesi ve dilin genel yapısına bağlıdır.
 - Çalışabilir kodun küçüklüğü ile çalışma hızı arasında doğrusal bir ilişki vardır.
 - C programları hızlı çalışır ve az yer kaplar.

Programlama Dillerinin Değerlendirme Ölçütleri

- Okunabilirlik
 - Kaynak kodun çabuk ve kolay bir biçimde algılanabilmesi anlamına gelir.
 - Okunabilirlik güncelleştirmeyi kolay kılar ve proje grubu halinde kodun üzerinde çalışılabilmesine olanak sağlar.
 - En iyi program kodu **anlaşılamayan ama çok zekice yazılmış kod değildir**.
 - En kolay okunabilen ve anlaşılabilen kod en iyi koddur.

"Babaannene iki dakikada açıklayamayacağın tek bir satır kodu bile programına ekleme, hatta babaannen Ada Lovelace olsa bile."

Anonim



Programlama Dillerinin Değerlendirme Ölçütleri

- Esneklik
 - Esneklik programlama dilinin programcıyı kısıtlamamasıdır.
 - Esnek dillerde birçok işlem, programcı için serbest bırakılmıştır.
 - Bu deneyimsiz programcılar için hata yapma riskini arttırır.
 - C esnek bir dil iken Java esnekliği çok kısıtlanmış bir dildir.

C dilinde karakter ve tamsayıların birbirine kolayca atanabilmesi

```
int main(){
    int x=97;
    char c=x;
    char b='b';
    int y=b;
    printf("%c\n",c);
    printf("%d\n",y);
    return 0;
}
```


Programlama Dillerinin Değerlendirme Ölçütleri

- Öğrenme Kolaylığı
 - Programlama dillerinin seviyesi arttıkça öğrenme kolaylığı artar.
 - Yüksek seviyeli dillerin popüler olması öğrenme kolaylığına bağlıdır.
 - C dili öğrenmesi zor bir dil iken Java aksine basittir.

Programlama Dillerinin Değerlendirme Ölçütleri

- Genel Amaçlılık

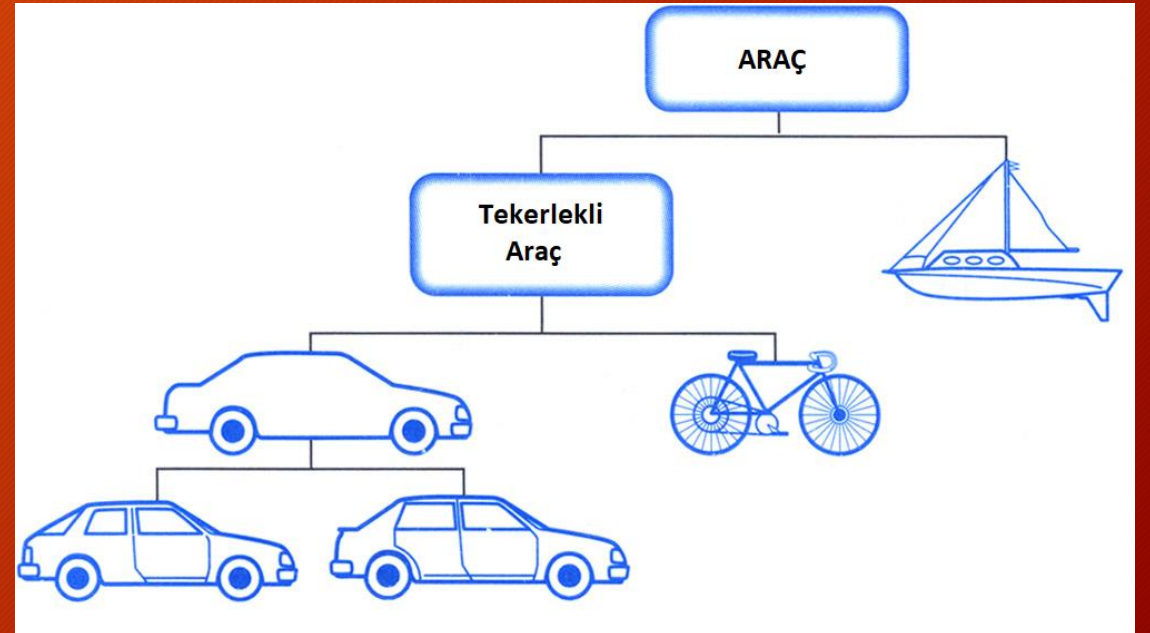
- Programlama dillerinin çok çeşitli uygulamalarda etkin olarak kullanılabilmesidir.
- Cobol dili ticari uygulamalarda etkin bir dil iken mühendislik uygulamalarında tercih edilmez.
- Java dilinin genel amaçlılığı yüksektir.

Programlama Dillerinin Değerlendirme Ölçütleri

- Yapısallık
 - Yapısallık bir programlama tekniğidir.
 - Bu tekniği kullanan dillerde bloklar halinde yazım ön plandadır.
 - Alt programlar yoğun olarak kullanılır.
 - C dili iyi bir örnektir.
 - Yapısal programlama 4 ana ilke üzerine kurulmuştur.
 - Böl ve Yönet
 - Veri Gizleme (lokal değişkenler)
 - Tek Giriş ve Çıkış
 - Döngüler ve Diğer Kontrol Yapıları

Programlama Dillerinin Değerlendirme Ölçütleri

- Nesne Yönelimlilik
 - Veri + Program = Nesne
 - Büyük programların yazılması için tasarlanmış bir tekniktir.
 - Üç temel üzerine kurulmuştur.
 - Kapsülleme
 - Çok Biçimlilik
 - Kalıtım



C Dili

- C dili ilk olarak Dennis Ritchie tarafından 1972 yılında Bell laboratuvarında geliştirilmiştir.
- C dili işletim sistemi dili olarak bilinir.
- C dili emir esaslı ve yapısal bir dildir.

Örnek bir C kodu

```
#include "stdio.h"
int main(){
    int yas;
    printf("Lutfen yasinizi girin:");
    scanf("%d",&yas);
    int dogumyili = 2017-yas;
    printf("Dogum Yiliniz:%d\n",dogumyili);
    return 0;
}
```

make Dosyası

- C dilinde gelişmiş programların tasarlanmasında birden fazla kaynak kod ve başlık dosyası kullanılabilmektedir. Bu durumda komutların tek tek el ile her seferinde komut satırına girilmesi zahmetli ve zaman alan bir iştir.
- Bu komutların bir dosyaya yazılıp dosyanın komut satırından çağırılması zamandan kazanç sağlayacaktır.
- Bu dosyanın ismine make dosyası (makefile) denilmektedir.

Örnek C kodunu derlemek için make dosyası

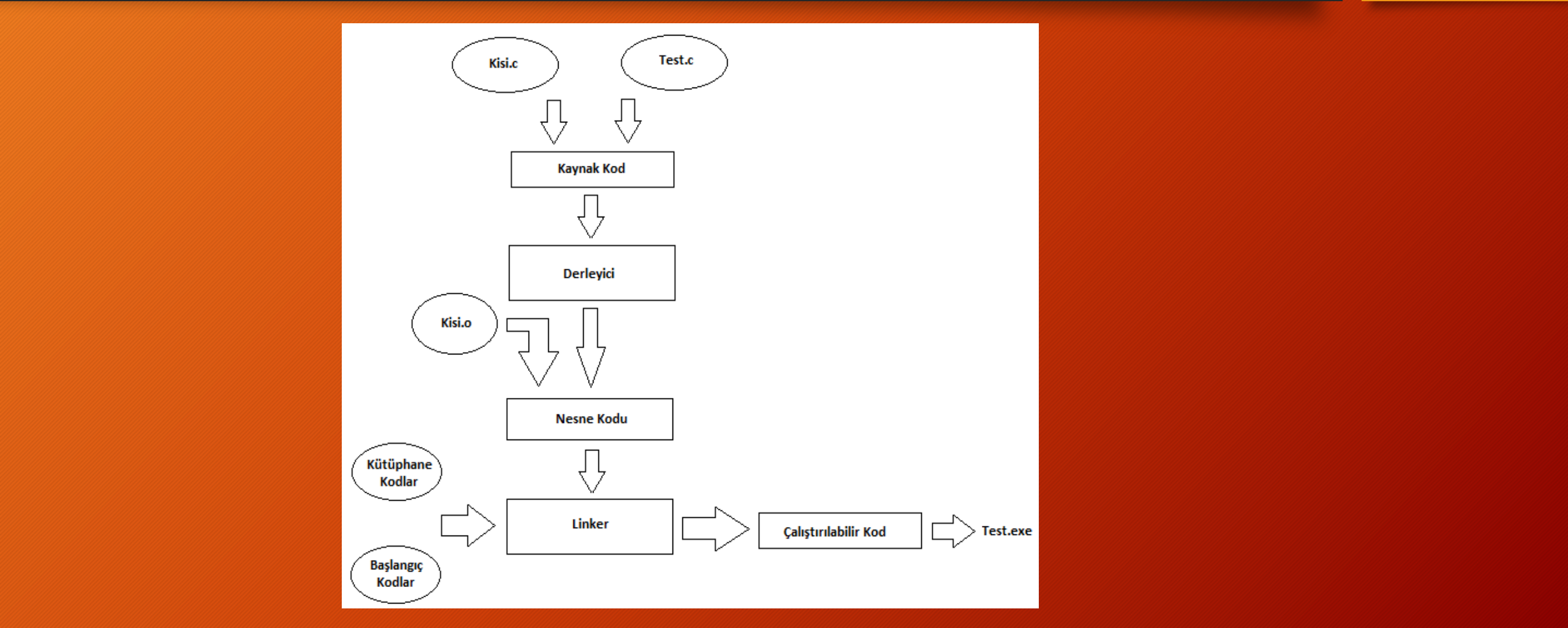
hepsi: derle calistir

derle:

```
gcc -I ./include/ -o ./lib/Kisi.o -c ./src/Kisi.c  
gcc -I ./include/ -o ./bin/Test ./lib/Kisi.o ./src/Test.c
```

calistir:

```
./bin/Test
```



Java Dili

- İlk olarak Oak ismiyle tasarlanmış ve gömülü sistemler için kullanılmıştır.
- Daha sonraları Java ismini alıp internet uygulama geliştirme için kullanılmıştır.
- Genel bir programlama dili olan Java her platformda kullanılabilir.
- Tamamen nesne yönelimlidir. En ufak program için yine Sınıf yazılmalıdır.

Lisp Dili

- Sembolik veri işleme için tasarlanan Lisp dili fonksiyonel programlama dilidir.
- Daha çok yapay zeka çalışmalarında kullanılmıştır.
- İki temel veri yapısı içerir bunlar **Atom** ve **Liste** dir.

Ekrana merhaba yazan Lisp fonksiyonu
<pre>(Defun MerhabaYaz() "Merhaba")</pre>

Prolog Dili

- 1970 yılında İngiliz ve Fransız ortaklığında geliştirilmiştir.
- Mantıksal programlama dili olan Prolog'ta bildirme esaslı bir yapı kullanılır.
- Prolog'u iki temel konsept oluşturur.
 - Olaylar
 - Doğru olan durumlardan oluşur.
 - Kurallar
 - Kuralları ifade etmek için önermelerden faydalanılır.
 - Atomik önermeler
 - Bileşik terimler

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016

Programlama Dillerinin Prensipleri

Hafta 2 - Programlama Dillerinin Tarihçesi ve Çeşitleri

Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

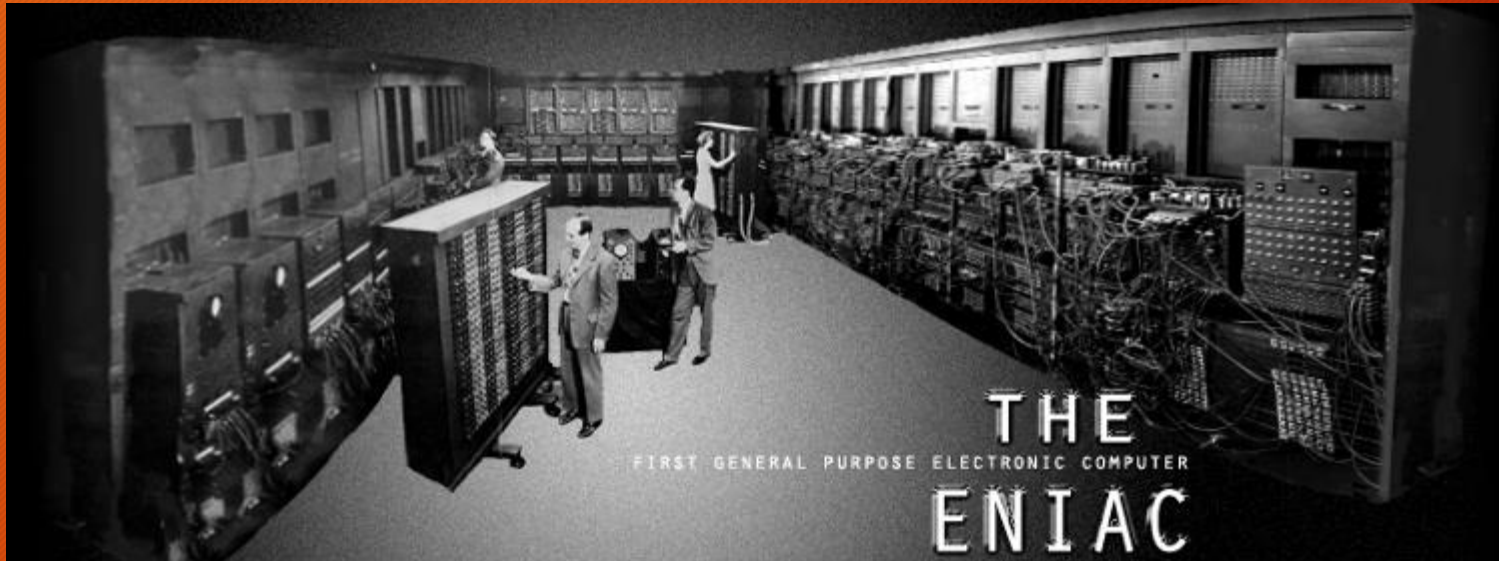
- Makine Dili
- Sembolik Diller
- Derleyicili Diller
- Yorumlayıcılı Diller
- Script Dilleri

İlk Yıllar

- 1800'lü yıllarda Charles Babbage, programlanabilir bilgisayar fikrini ilk ortaya attı.
- Bunlar fiziksel hareketlerden ibaretti elektriksel sinyallere dönüşmesi ENIAC ile 1942 yılını bulacaktı.
- Bu yıllarda programlama denilince akla gelen, çözülecek probleme ilişkin bir devre tasarlamak anlaşılmaktaydı.
- Bu yıllarda sadece Elektronik Müh. Programlama yapabilmekteydi.

Makine Dili

- Sadece ikili sayı sisteminde oluşmaktadır. (0-1)
- ENIAC sadece matematiksel işlemler yaabilen (0-1) ile çalışan bir makine dilidir. Aynı zamanda ENIAC makinenin adıdır.



Sembolik Dil

- 1950'li yıllarda Sembolik dil (Assembly) geliştirildi.
- İkili kodlarla program yazmak oldukça zor olduğu için ikili kodlar sembollerle ifade edilmiştir.
- Komutların adlandırılması ve akılda kalması kolaylaştırılmıştır.
- Tamamen donanıma bağlı düşük düzeyli bir programlama dilidir.

Sembolik Dil

- Burada bellekten okuma yazma yerine çok daha hızlı olması açısından register'lar kullanılır.
- Sembolik diller bilgisayar kullanımını hızla arttırmıştır.
- Ancak çok basit işlemler için bile birçok komut gerekmektedir.
- Ayrıca sembolik diller her seferinde makine diline çevrilip öyle çalıştırılıyordu. Bu işlem program hızını 30 kat yavaşlatıyordu.

Swap işlemi

mul	\$2, \$5, 4
add	\$2, \$4, \$2
lw	\$15, 0(\$2)
lw	\$16, 4(\$2)
sw	\$16, 0(\$2)
sw	\$15, 4(\$2)
jr	\$31

Assembler

```
000000001010000100000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
100011001111001000000000000000100
10101100111100100000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

Derleyici Fikri

- Sembolik dil bir yorumlayıcıydı.
- Her seferinde makine diline çevrilip öyle çalıştırılıyordu.
- Grace Hopper, bu problemin çözümü için derleyici fikrini ortaya attı.
- Program kodu bir kez derlenip makine diline çevrilecek ve bir daha bu işleme gerek kalmayacaktı.

Fortran Dili

- Fortran (FORmul TRANslating System) 1954 yılında IBM firmasında John Backus tarafından geliştirildi.
- Do deyimleri, G/Ç deyimleri ve atama deyimleri içeriyordu.
- Matematiksel denklemlerin çözümü amaçlanmıştı.
- Çalışma sırasında veri tiplere ifadeleri ve bellek tahsisi yoktu.
- Fortran ifadeleri İngilizce kelimelerden oluşuyordu ve sembolik dile göre anlaşılması çok kolaydı.

Fortran Dili

- Fortran diline bir derleyici yazılıp piyasaya sunulması 1957 yılını bulacaktı.
- Fortran derleyicisi bir teyp biriminde saklanıyordu.
- Fortran'ın daha sonra birçok sürümü yayınlanmıştır.



Fortran'ın Sürümleri

- Fortran I
 - Taşınabilirlik yönünden oldukça kötü bir dildir.
- Fortran 66
 - Karakter türü verileri işlemede çok kısıtlı. Yapısal programlama desteklenmiyor.
- Fortran 77
 - ANSI karakter türü verileri işleyebiliyor.
 - Yapısal programlamayı destekliyor.

Fortran 90 ve 95

- Bu yıllarda C dili çok iddali ve popüler olmaya başlamıştı.
- Dolayısıyla Fortran 90'da C'deki birçok özellik eklenmeye çalışıldı.
- Fortran 90'da pointer, özyineleme, bit düzeyinde işlem ve dizi yapıları daha kullanılabilir hale getirildi.
- Fortran 95'in ise temel hedefi taşınabilirliğin mükemmel hale getirilmesi olmuştur.
- Fortran 95'te nesneye yönelik programlama özellikleri de eklenmeye çalışılmıştır.

Fonksiyonel Paradigma

- Fonksiyonel paradigma ile ilk 1958 yılında tanışılmıştır.
- Fonksiyonel programlama dili olarak geliştirilen Lisp daha önce tanıtılan dillerden çok farklıydı.
- Lisp daha çok yapay zeka uygulamaları için kullanılmaktaydı.
- Atom ve Liste adı verilen iki veri yapısına sahipti.
- Lisp'in daha sonraki iki sürümü
 - Scheme (1970)
 - Common Lisp (1984)

Algol Dili

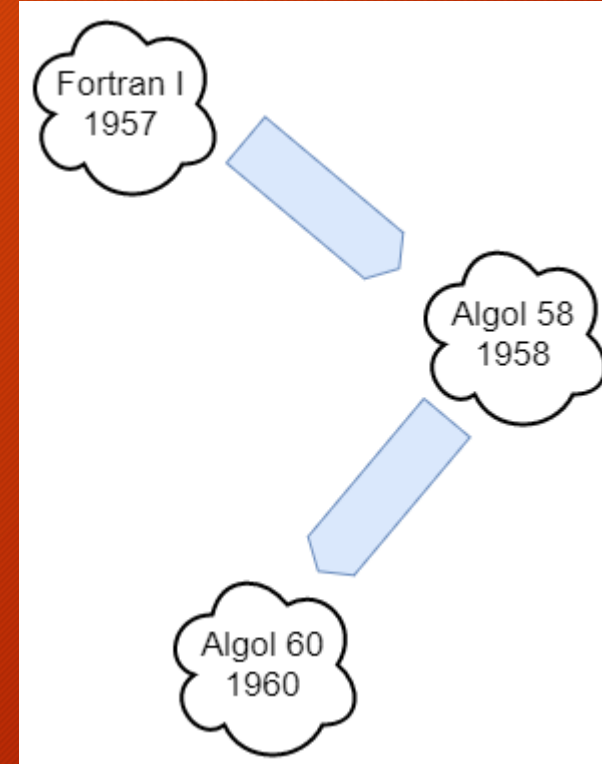
- 1958 yılında Avrupalı ve Amerikalı bir komisyonun Fortran I'den esinlenerek Zürih'teki çalışmaları sonucu yüksek seviyeli dil olan Algol (ALGOritmic Language) geliştirmişlerdir.
- Bu dile ilk olarak Algol 58 ismi verilmiştir.
- Algol 60
 - Algol 58'e eklemeler yapılmıştır.
 - Daha matematiksel notasyonlara yakın ve kolay okunabilir bir dil olmuştur.
 - Makine diline kolaylıkla çevrilebilen bir dil olmuştur.

Algol 60'ın Başarılı ve Eksik Yönleri

Başarılı Yönleri	Eksik Yönleri
Yapısal programlama tekniği benimsenmiştir	Aşırı esnek olmasından dolayı anlaşılabilirliği düşmüştür.
Pass by value ve pass by name desteklenmektedir.	Yürütmede verimsizliğe götürecek esnek yapısı bulunmaktadır.
Özyineleme desteklenmektedir.	I/O ifadelerinde yetersizlik ya da zayıflık bulunmaktadır.
Stack-dinamik dizilere izin verilmiştir.	

Algol Dilinin Soy Ağacı

- Bilim adamlarının eğitim ve araştırma aracı olarak Algol dilini kullanmasına rağmen IBM tarafından desteklenmemiştir.
- Çünkü IBM bu sırada Fortran'ı desteklemekteydi.
- O zamanlar için IBM'in bilişim sektörünün %80'ine sahip olduğu düşünüldüğünde bu durum Algol için bir dezavantaj olmuştur.

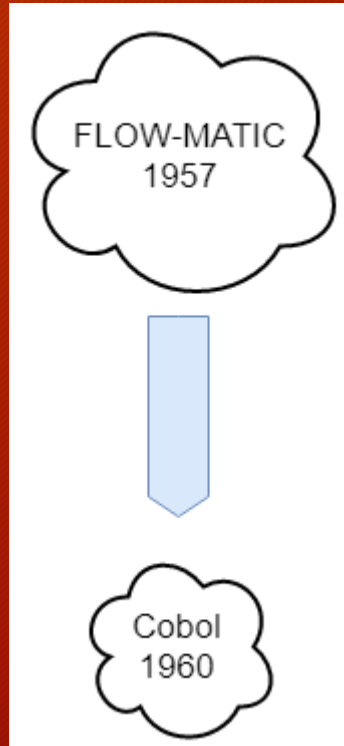


Algol 68

- Algol 60'tan yaklaşık 8 sene sonra geliştirilmiştir.
- Kullanıcı tanımlı veri tiplerini destekleyen ilk programlama dilidir.
- Aynı zamanda dinamik dizi kavramına izin veren ilk dildir.
- Algol 68 öğrenilmesi oldukça zor olan bir gramer ve dil yapısına sahipti.

Cobol

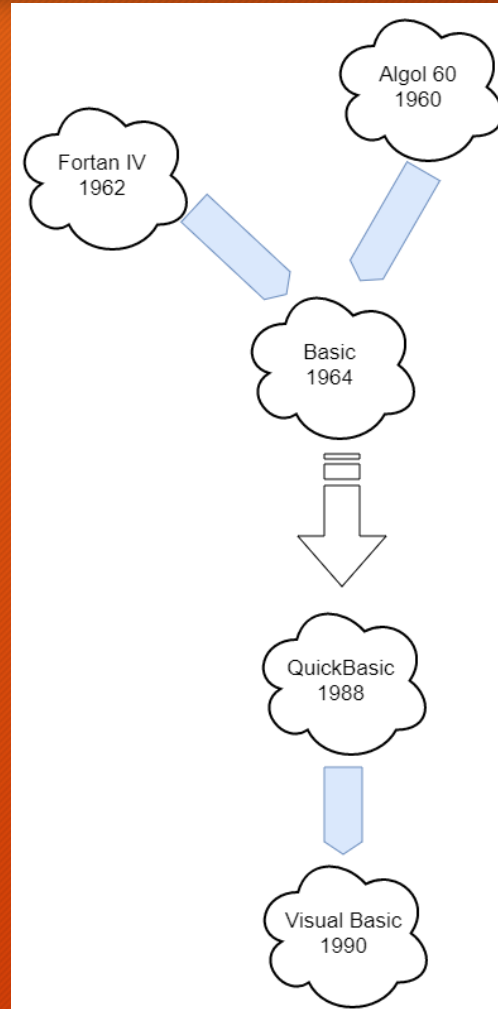
- 1959 yılının mayıs ayında Cobol (Common Business Oriented Language) tanıtılmıştır.
- Bu dil savunma sektörünün çalışmaları sonucu ortaya çıkmıştır.
- İşletmelere yönelik bir dildir.
- Bilgisayar büyük miktarda bilgi giriş-çıkışının yapıldığı uygulamalar için geliştirilmiştir.
- Hiyerarşik veri yapıları ve yan anlamlı isimlerin görüldüğü ilk dildir.
- Fonksiyonlar desteklenmemektedir.
- 1990'lı yıllarda nesne yönelimli versiyonu üretilmiştir.



Basic

- 1964 yılında tanıtılmıştır.
- Öğrenilmesi oldukça kolaydır.
- Dolayısıyla daha çok eğitim amaçlı kullanılmıştır.
- Uzaktaki bir bilgisayarla bağlantı kuran ilk programlama dilidir.
- İlk başlarda sadece 14 komuta sahipti ve tek veri tipi vardı.
 - LET, PRINT, GOTO, ...
 - number veri tipi (kayan noktalı ve tamsayı)

Basic Soy Ağacı



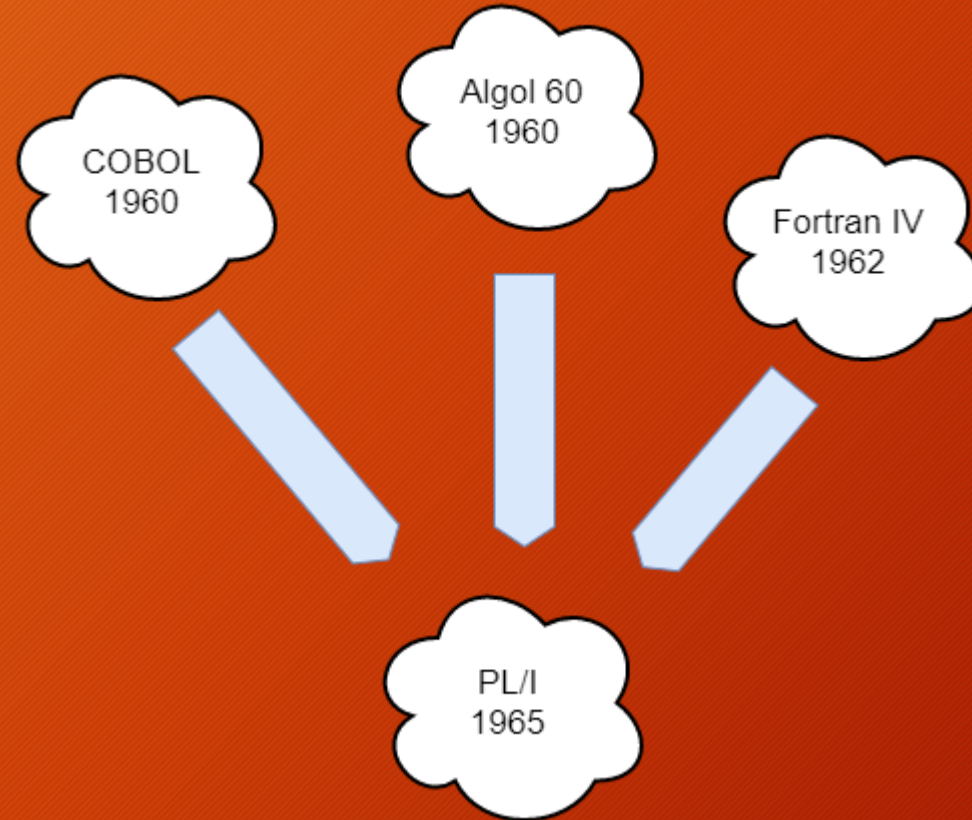
Basic Dilinin Genel Özellikleri

- Kolay ve genel amaçlı bir dil
- Açık ve anlaşılır hata mesajlarına sahip
- Küçük boyutlu programları hızlı bir şekilde çalıştırabilir.
- Kullanımı için donanım bilgisine sahip olmaya gerek yok.
- Kullanıcıyı işletim sistemi ayrıntılarından dahi soyutlayabilmektedir.
- 1990 yılında nesne yönelimli uyarlama olan visual basic sunulmuştur.

PL/I (Programming Language One)

- 1965 yılında bilimsel ve işletme problemlerine çözüm sağlayabilmek için geliştirilmiştir.
- Floating-point ve desimal veri tiplerini desteklemektedir.
- Eşzamanlı çalışan alt programlara izin vermektedir.
- Özyinelemeyi desteklemektedir.
- Pointer kullanımına izin vermektedir.
- Hafıza gereksinimi yüzünden karmaşık ve tasarım yönünden iyi değildir.

PL/I Soy Ağacı



Simula 67

- İlk olarak simülasyon için tasarlanmıştır.
- İlk nesne yönelimli dildir.
- Algol 60'ın genişletilmiş versiyonudur.
- Veri soyutlamasına imkan veren sınıfları desteklemektedir.

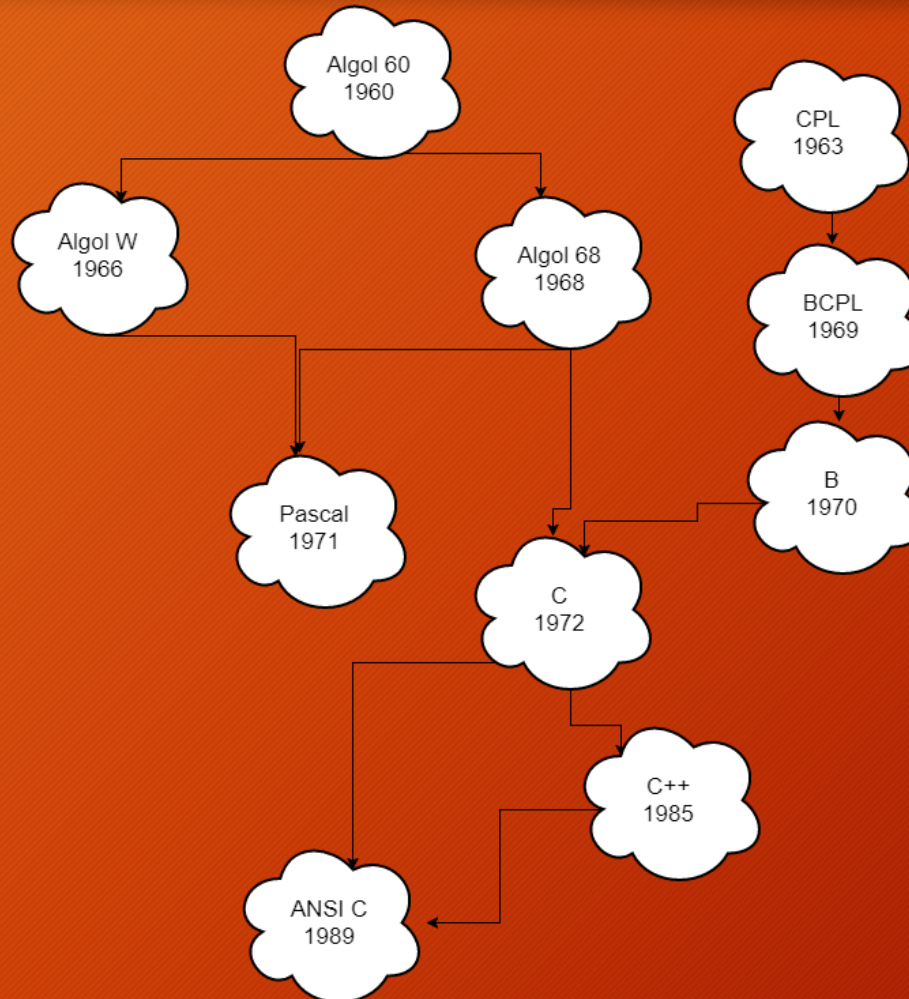
Simula 67 Soy Ağacı



ALGOL'un Torunları

- Pascal
 - 1971 yılında tanıtıldı.
 - Basit ve okunabilir bir dil olduğu için ilk önceleri eğitim dili olarak kullanıldı.
 - Fortran ve C diline göre güvenli bir dildir.
- C Programlama Dili
 - 1972 yılında tanıtıldı.
 - Aktif bellek erişimi, taşınabilir özelliği güçlü yanlarıdır.
 - Bir sistem programlama dilidir.
 - Yeterli derecede kontrol ifadelerine sahiptir.
 - Tip kontrolü yetersizdir.

Pascal ve C Dillerinin Soy Ağacı



Modula

- Güçlü bir tip ayrımı ve tip kontrolü mekanizmasına sahiptir.
- Dinamik dizi kullanılabilir.
- Daha sonra geliştirilen Modula 2 yazımı daha esnek bir dildir.
- Soyut veri tipini desteklemekte fakat kalıtım olmadığı için nesne yönelimli bir dil değildir.

Prolog

- 1972 yılında geliştirilmiştir.
- Çok yüksek seviyeli programlama dilinin ilk örneğidir.
- Dil kurallar ve önermelerden oluşmaktadır.
- Yapay zeka alanlarında kullanılmıştır.

Ada

- ABD savunma bakanlığını bir çalışması sonucu ortaya çıkmıştır.
- Programlama dilleri tarihindeki en geniş tasarım çalışmasıdır.
- Blok yapılı, nesne yönelimli ve eş zamanlılığı destekleyen bir dildir.
- Kalıtım ve çok biçimlilik özellikleri eklenerek Ada 95 sürümü tanıtılmıştır.

Ada Dili Soy Ağacı



Smalltalk

- 1970'li yıllarda Alan Kay ve grubu tarafından geliştirildi.
- Ana amacı okunması ve geliştirilmesi kolay bir dil tasarlamaktı.
- Tamamen nesne yönelimli bir programlama dilidir.



C++

- Emir esaslı ve nesne yönelimli programlama paradigmasının birleşiminden oluşur.
- C++ dili C ve Smalltalk'ın birleşimidir.
- Çoklu sınıf kalıtımı desteklenmektedir.
- Operatör ve fonksiyon overloading desteklenmektedir.
- Şablon sınıf ve metot desteği vardır.
- Aktif bellek yönetimini destekler.
- Java dilinden daha az güvenli bir özelliğe sahiptir.

Eiffel Dili

- Neredeyse bir C++ dili olarak görülebilir.
- Emir esaslı ve nesne yönelimli programlama paradigmalarını destekler.
- C++'tan daha küçük ve basittir.

Delphi

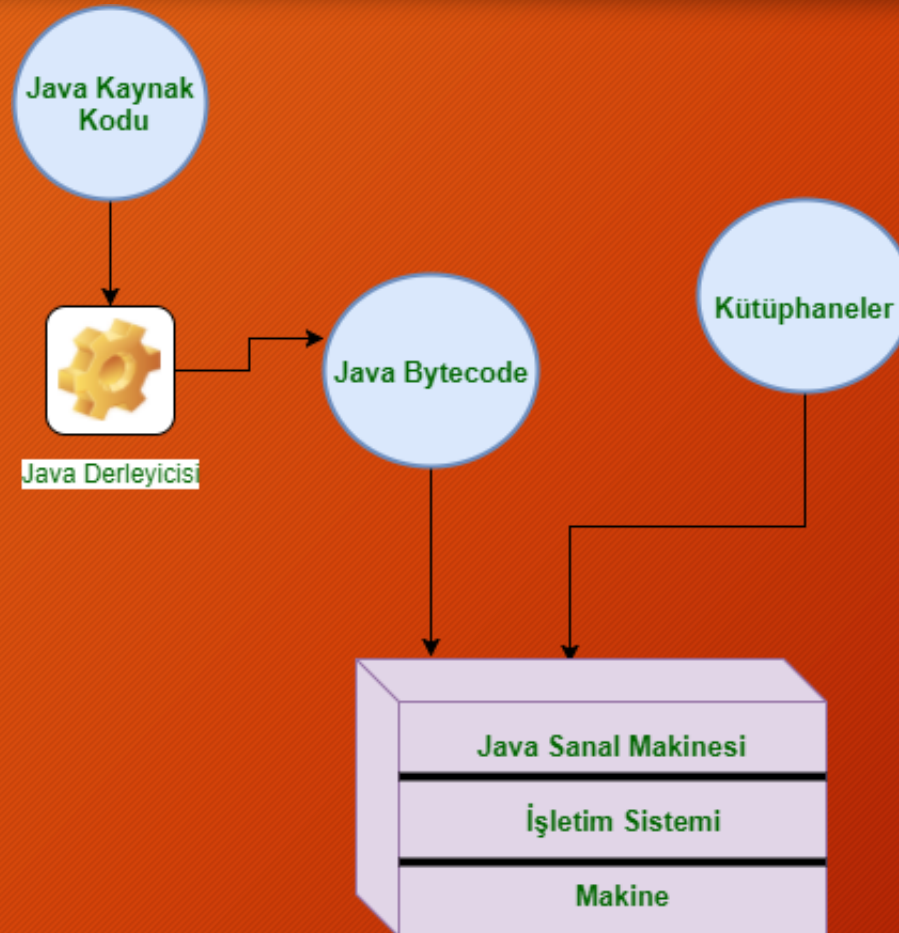
- Emir esaslı ve nesne yönelimli programlama paradigması başarılı bir biçimde birleştirilmiştir.
- Pascal'dan esinlenerek geliştirilmiştir.
- Dizi elemanlarının kontrolü, pointer aritmetiği ve tip zorlamada C/C++'tan daha güvenlidir.
- Delphi daha iyi ve kolay yazılım geliştirmek için GUI sağlamaktadır.

Delphi'de dizi tanımı ve kullanımı	
var	
	sayilar : array[5..20] of string;
	i : Integer;
begin	
	for i := 5 to 20 do
	sayilar[i] := IntToStr(i * 5);
end;	

Java

- C++ temel alınarak daha küçük, basit ve daha güvenilir bir programlama dili geliştirilmiştir.
- Tamamen taşınabilir ve nesneye yönelik bir programlama dilidir.
- Java dilinde her şey sınıflardan oluşur.
- Eski Java sürümlerinde şablon yapıları yoktur. Bu ihtiyaç Object sınıfı kullanılarak giderilebilmekteydi.
- Java dilinde otomatik çöp toplayıcı mekanizması vardır.

Java Sanal Makinesi



Script Dilleri

- Bir dizi komutun bir dosya içerisine konulması ve dosya çağrıldığında komutların çalıştırılması prensibine dayanır.
- İlk script dili David Korn tarafından geliştirilen ksh olarak gösterilebilir.
- Daha sonra bu dil awk ve Perl gibi dillerin doğmasına ön ayak olacaktır.
- Çalışma şekli olarak iki türlü script söylenebilir.
 - İstemci taraflı
 - Sunucu taraflı

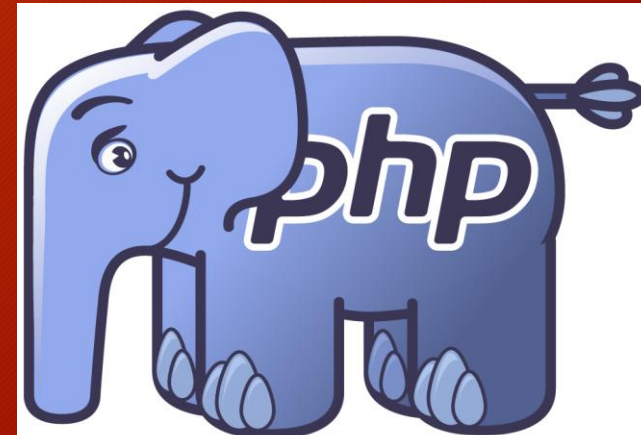
Javascript

- İstemci taraflı script diline örnektir.
- Netscape ve Sun'ın ortak çalışması sonucu 1995'te tanıtılmıştır.
- Java'nın kuvvetli tip kontrolü bu dilde daha esnektir.



Php

- Sunucu tabanlı bir script dilidir.
- Rasmus Lerdorf tarafından 1994 yılında geliştirilmiştir.
- Kodlar sunucu tarafında yorumlanır.
- Php dizi yapısı Javascript ve Perl dizi yapılarının bir bileşenidir.



Python Dili

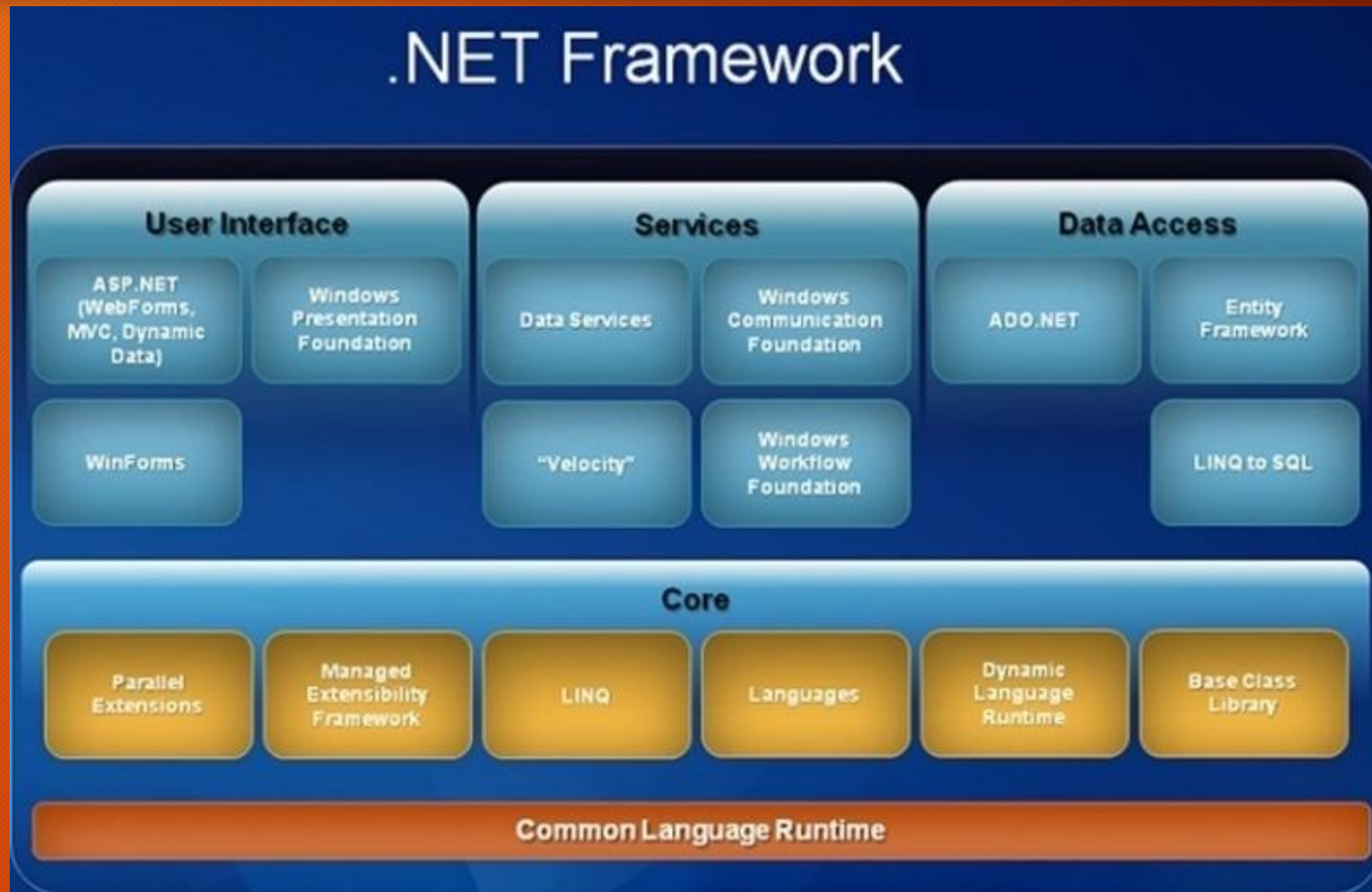
- 1991 yılında Guido van Rossum tarafından geliştirildi.
- Yüksek seviyeli ve genel amaçlı bir programlama dilidir.
- Başlarda Amoeba işletim sisteminde çalışacak şekilde tasarlanan python dili daha sonra her platformda çalışabilecek hale gelmiştir.
- Başarılı bir istisnai durum yönetimi olan python dili yorumlama şeklinde çalışır.
- Bir çok veri yapısı desteği yanında nesne yönelimli bir dildir.
- Devrim yaratan versiyonu 2000 yılında tanıtılan Python 2.0 olarak bilinir.
- Çöp toplayıcıya sahiptir.



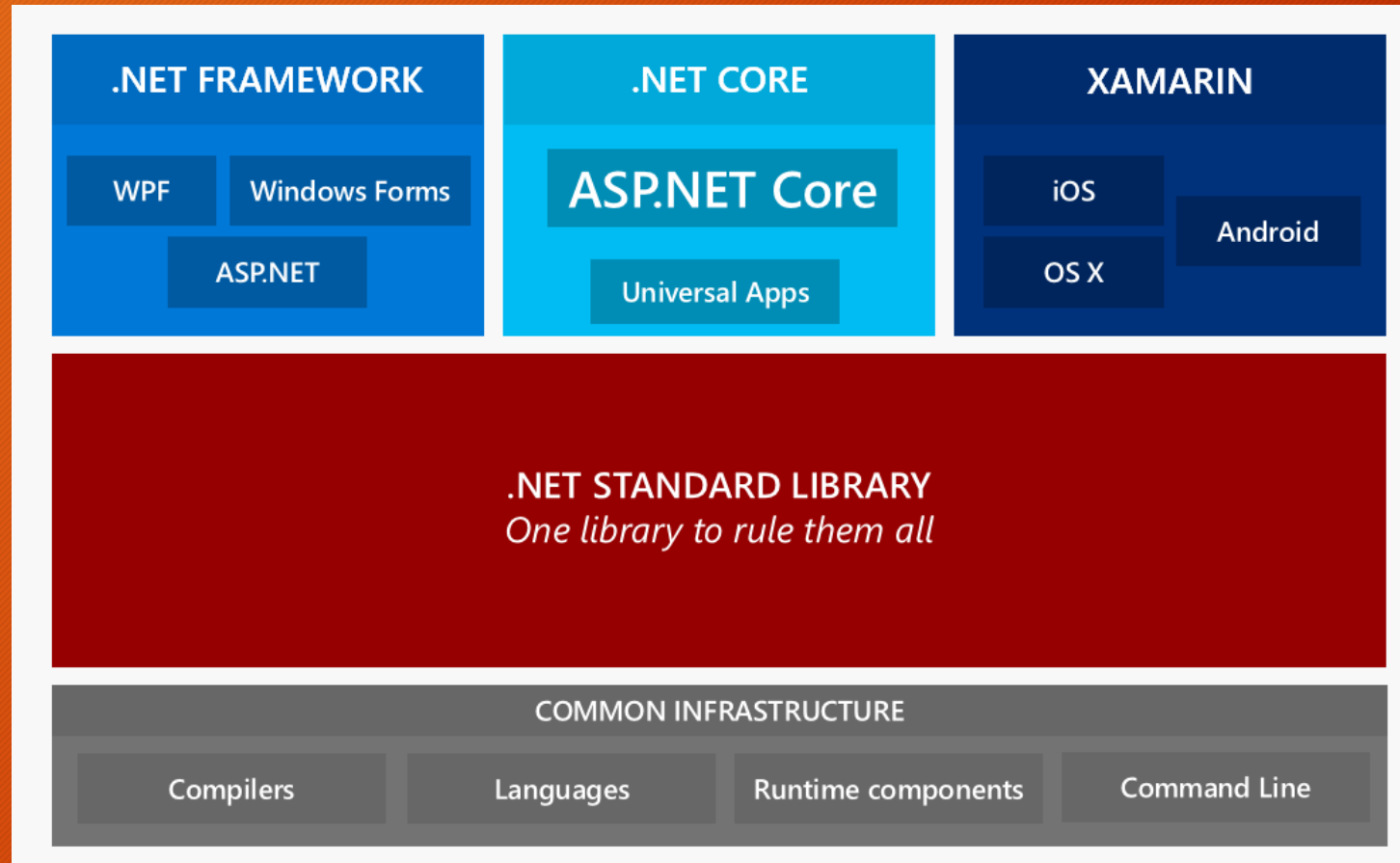
C# Dili

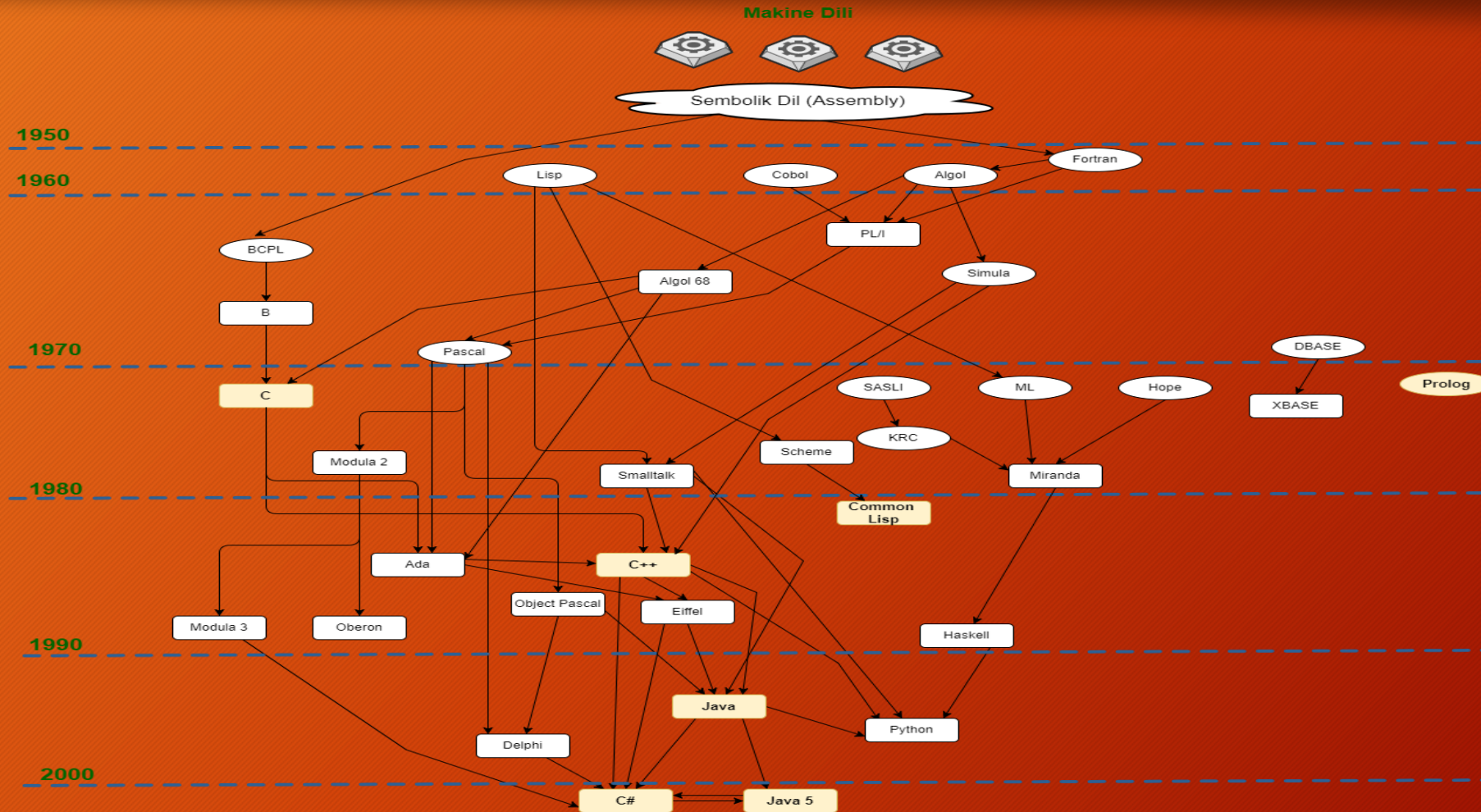
- C ve C++ dil ailesinin ilk bileşen yönelimli (Component-oriented) dilidir.
- Common Language Runtime (CLR)
 - Bir yürütme motoru
 - Bir çöp toplayıcı
 - Anında derleme
 - Güvenlik sistemi
 - Zengin bir sınıf çerçevesi (.NET Framework)
- CLR birçok dil desteğine kadar herşey için tasarlanmıştır.
- Otomatik bellek yönetimi kullanır.

.NET Framework



.NET CORE





Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016

Programlama Dillerinin Prensipleri

Hafta 7 - Nesne Yönelimli Programlama

Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

- Nesnelerin Görünüşleri
- Nesneye Dayalı Düşünme
- Sınıf Hiyerarşisi
- this terimi
- Nitelikler
- İç içe Sınıflar
- Yıkıcı Metot
- Erişim Niteleyicileri
- Kalıtım
- Overload ve Override

Tanım

- Nesneye dayalı programlama kompleks sistemleri yapılandırma ve yönetmeye imkan sağlamaktadır.
- Günlük hayatımızda kullandığımız şeylerin taklit edilmesi esasına dayanılır.
- Nesnelerin tanımlanmasını ve kullanımını destekleyen dillere nesne yönelimli dil denir.
- Tüm nesneye yönelik diller, programlama dilleri literatüründe sınıf ve alt sınıf kavramını ilk defa tanıtan SIMULA67 diline dayanmaktadır.

Nesnelerin Görünüşleri

- Bir problemin çözümünde bir nesne herhangi bir varlık olarak ifade edilebilir.

Harici Görünüş

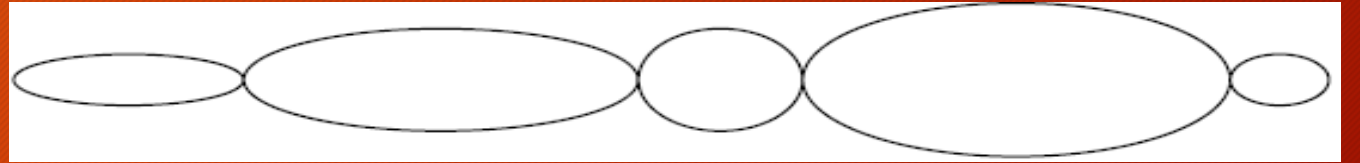
Kargo Takip Sistemi

- Kargo
- Müşteri
- Nakliye

Bu kullanılabilecek varlıklardır. Bunlar nesnelerin harici görünüşüdür.

Farklı Özelliklere Sahip Aynı Tür Nesneler

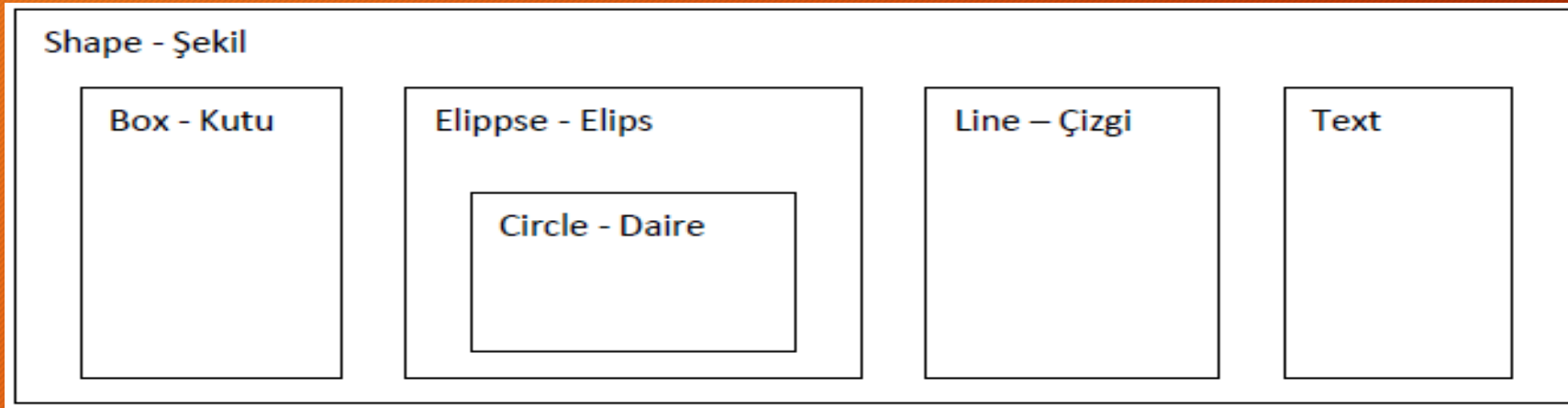
- Genişlik ve yükseklikleri farklı fakat aynı elips nesneleri



Nesneye Dayalı Düşünme

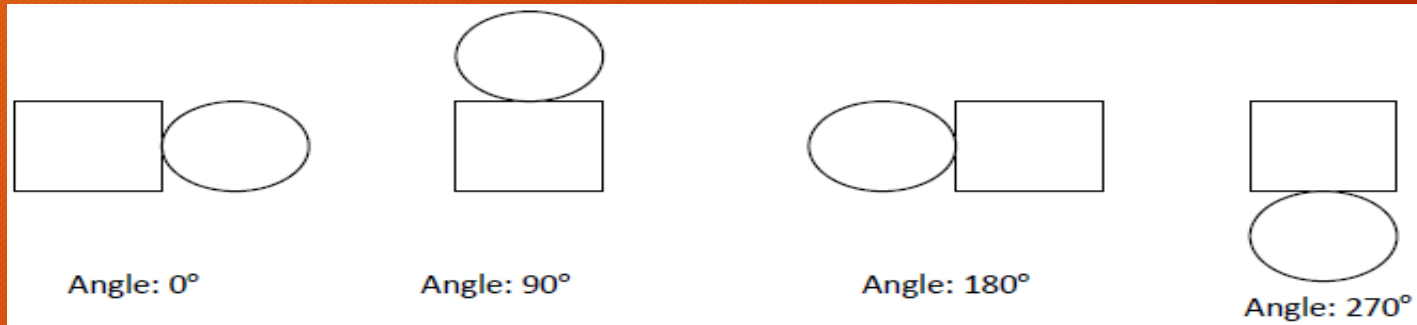
- Object (nesne) : Veriler ve işlemler topluluğu
- Class (sınıf) : Nesneler kümesinin bir tanımı
- Subclass (altsınıf) : Bir sınıfın ilave özelliklere sahip bir kümesi
- Instance (örnek) : Bir sınıfın bir nesnesi için kullanılan teknik bir terim
- Method (metod) : Bir işlemi yürüten bir altprogram içeriği
- Message (mesaj) : Bir metodu işlemek için bir istek, bir alt programın çağırılması

Sınıf Hiyerarşisi



- Bir sınıfın içine girmiş sınıf alt sınıftır. Tersisi süper sınıf veya üst sınıftır.
- Bir nesneye gelen bir mesaj, bir altprogramın çağrılmasına benzer.

Nesne Türerilmesine Örnekler



Sınıf

- Kapsüllemeyi, veri gizlemeyi ve çok biçimliliği mümkün hale getiren yapı.
- Kullanıcı tanımlı bir türdür.
- Veri ve metotları içerir, dolayısıyla mesaj gönderilip cevap alınabilir.

Nesne Terimi



Nesnenin Oluşturulması

- Sınıftan türetilecek bir nesnede içerisinde bulunacak elemanlar ilk değerlerini almaları gerekir.
- Bu durumda nesne oluşma aşamasında bir metot çağrılmalıdır.
- Bu fonksiyon kurucu veya yapıcı (constructor) fonksiyondur.

```
// Java  Öğrenci o1 = new Öğrenci("Ahmet Almaz",159);
class Öğrenci{
    private String isim;
    private int numara;

    public Öğrenci(String ism,int nmr){
        isim=ism;
        numara = nmr;
    }
}
```

```
//C++  Öğrenci o1("Ahmet Almaz",159);
class Öğrenci{
    private:
        string isim;
        int numara;
    public:
        Öğrenci(String ism,int nmr){
            isim=ism;
            numara = nmr;
        }
};
```


this Terimi

- Sınıf hiyerarşisinde, o anda oluşturulan nesneyi ifade etmek için kullanılır.
- Bazı durumlarda kullanımı gereklidir.
- Ayrıca okunabilirliği arttırmak için de kullanılmaktadır.

this teriminin kullanımı

```
class Kisi{  
    public String isim;  
    ...  
  
    public Kisi(String isim){  
        isim=isim;  
        yas=0;  
        boy=20;  
        kilo=4;  
    }  
    ...  
}  
  
// doğrusu  
public Kisi(String isim){  
    this.isim=isim;  
    yas=0;  
    boy=20;  
    kilo=4;  
}
```

Nitelik Tanımı (Property)

- Nitelikler bazı programlama dillerinde desteklenmektedirler.
- Amaç sınıftaki bir veya birden fazla alt alanı okuma ve yazma yönünden korumaya almak veya yönetebilmektir.
- Nitelik desteklemeyen programlama dillerinde bu metotlar yardımıyla yapılır.

```
class Kisi
{
    private double kilo;
    private double kalori;

    public Kisi()
    {
        kilo = 4;
        kalori = 0;
    }
}
```

// C# dili

```
public double Kalori
{
    get
    {
        return kalori;
    }
    set
    {
        kalori = value;
        if (kalori > 1000)
            kilo += (kalori / 1000);
    }
}

public double Kilo
{
    get
    {
        return kilo;
    }
}
```

Yıkıcı Metotlar

- Çöp toplayıcı araçlarına sahip diller için dikkat edilmesine gerek olmayan metotlardır.
- Fakat söz konusu C++ gibi bir dil ise Heap bellek bölgesinde oluşturulan nesnelerin belleğe geri iadesi gerekeceğinden yıkıcı metotların önemi büyüktür.

Yıkıcı Metotlar

- C++ dilinde programcı tarafından çağrılabilen yıkıcı metot java ve C#, Python gibi dillerde çöp toplayıcı tarafından çağrılır.

```
//Java
public class Arac{
    @Override
    protected void finalize() throws Throwable {
        // Çöp toplayıcı tarafından çağrılır.
    }
}
```

```
// C#
class Arac : IDisposable
{
    ~Arac()
    {
        // Çöp toplayıcı tarafından
    }

    public void Dispose()
    {
        // Programcı çağırabilir
        // Nesnenin yok edilmesini garantilemez
    }
}
```

```
// Python
class Arac:
    def __del__(self):
        # Çöp toplayıcı tarafından
```

Erişim Niteleyicileri

- Sınıflarda bilgi gizleme, dış kullanıma açma, kalıtım için kullanma gibi işlemler için erişim niteleyicileri kullanılır.

Varsayılan: Bazı dillerde

C++ (struct) public, C++(class) private

C# private

Java aynı pakette ise sınıf dışından erişilebilir.

private: Sadece sınıf içerisinden erişilebilir.

protected: Sınıf dışından sadece kalıtım yolu ile erişilebilir.

public: Sınıf içi ve dışında erişilebilir.

Erişim Niteleyicileri

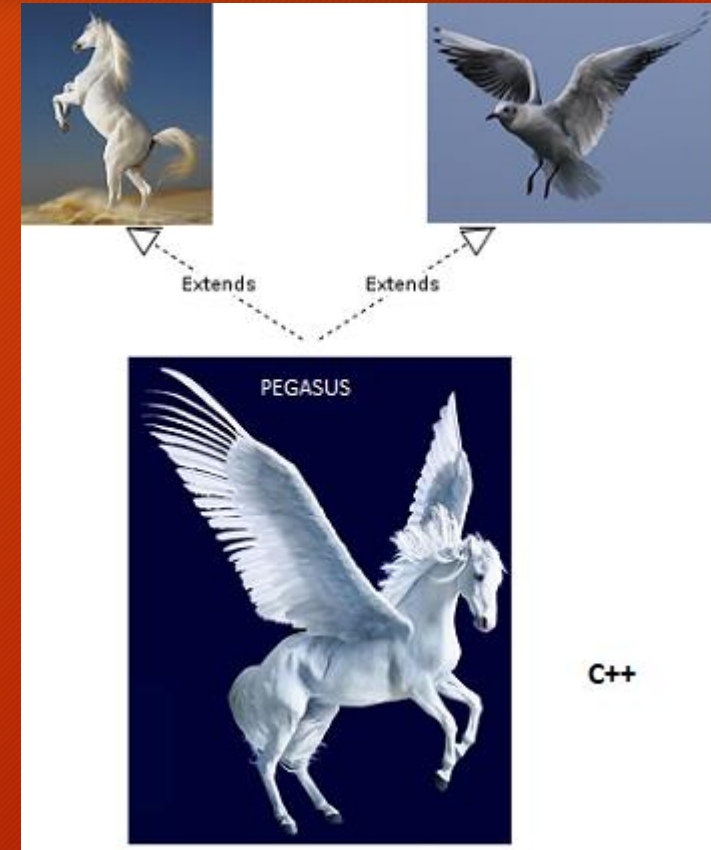
- Bilgi gizleme iki çeşit değişikliğe imkan sağlar.
 - Yürütme (implementation) Değişiklikleri
 - Eğer bir nesne ile olan bütün etkileşimler o nesnenin arayüzü üzerinden yapılıyorsa o zaman arayüzün arkasında gizlenen algoritmalar ve veri yapıları yürütme yardımıyla değiştirilebilir.
 - Kalıtım Değişiklikleri
 - Bir üst sınıfa olan bütün etkileşimler arayüzler üzerinden olursa program alt sınıfların eklenmesi ile genişletilebilir.

Kalıtım

- Birçok tür tamamen bağımsız olmayacak ve bazı özellikleri başka türlere benzeyecektir.
- Bu durumda bu özelliklerin bir üst sınıftan alınması geliştirme sürecini hızlandırır.
- Kalıtım alınan sınıfa üst sınıf denir ve programlama dillerinde yukarıya mesaj gönderme şekli;
 - Java `super`
 - C# `base`
 - C++ Sınıf adı ile

Kalıtım

- C++ gibi bazı dillerde çoklu sınıf kalıtımı desteklenir.
- Java ve C# dillerinde çoklu arayüz kalıtımı desteklenmektedir.
- Çok biçimlilik yine kalıtım ile desteklenmektedir.



Çoklu Kalıtım ve Diamond Problemi

```
class Canli{
public:
    double kilo;
};

class Kus : public Canli{
public:
    Kus(){
        kilo=0.25;
    }
    void YemekYe(double kalori){
        kilo += (kalori/10000);
    }
    double Kilo(){
        return kilo;
    }
};

class At : public Canli{
public:
    At(){
        kilo=100;
    }
    void YemekYe(double kalori){
        kilo += (kalori/1000);
    }
    double Kilo(){
        return kilo;
    }
};

class Pegasus : public At, public Kus{
public:
    Pegasus(){

    }
};

int main(){
    Pegasus *p = new Pegasus();
    p->YemekYe(1500);
    cout<<p->Kilo();
    delete p;
    return 0;
}
```


Overload (Aşırı Yükleme) ve Override (Ezme) Terimleri

- Overload bir metoda yeni anlamlar kazandırarak çok biçimliliği destekler.
- Override ise metodu yeniden tanımlamaktır. Dolayısıyla eski metodu ezmiş olacaktır.

C# dilinde operatörlerin aşırı yüklenmesi	C#'ta ilkel türe benzer kullanım
<pre>class Sayi { public int deger; public Sayi(int dgr) { deger = dgr; } public static Sayi operator +(Sayi s1, Sayi s2) { return new Sayi(s1.deger+s2.deger); } public override string ToString() { return this.deger.ToString(); } }</pre>	<pre>static void Main(string[] args) { Sayi x = new Sayi(10); Sayi y = new Sayi(5); Console.WriteLine(x+y); Console.ReadKey(); } static void Main(string[] args) { int x = 10, y = 5; Console.WriteLine(x+y); Console.ReadKey(); }</pre>

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016

Programlama Dillerinin Prensipleri

Hafta 9 - Nesne Yönelimli Programlama - 2

Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

- Arayüzler
- Çoklu Sınıf ve Arayüz Kalıtları
- namespace ve Paketler
- Soyut Sınıflar
- Kalıtım Hiyerarşisi
- Object Veri Türü
- Generic - Şablon Yapılar
- Friend Erişim Niteleyicisi

Arayüz Tanımı

- Arayüz kullanımı aslında bir sözleşme imzalamaktır.
- Bir arayüzden kalıtım alan bir sınıf o arayüzü gerçekleştireceğini vadediyor demektir.
- Arayüzler yardımıyla kullanılabilirlik dış dünyaya rahatlıkla açılabilir.
- Java ve C#'ta arayüzler alan içeremez sadece public metot tanımı içerebilirler.
- Java ve C# dilinde direk desteği bulunan arayüzler, C++ dilinde soyut sınıflar kullanılarak gerçekleştirilir.

```
public interface MuzikCalar {  
    public void Oynat(String dosyaTuru,String dosyaAdi);  
}
```

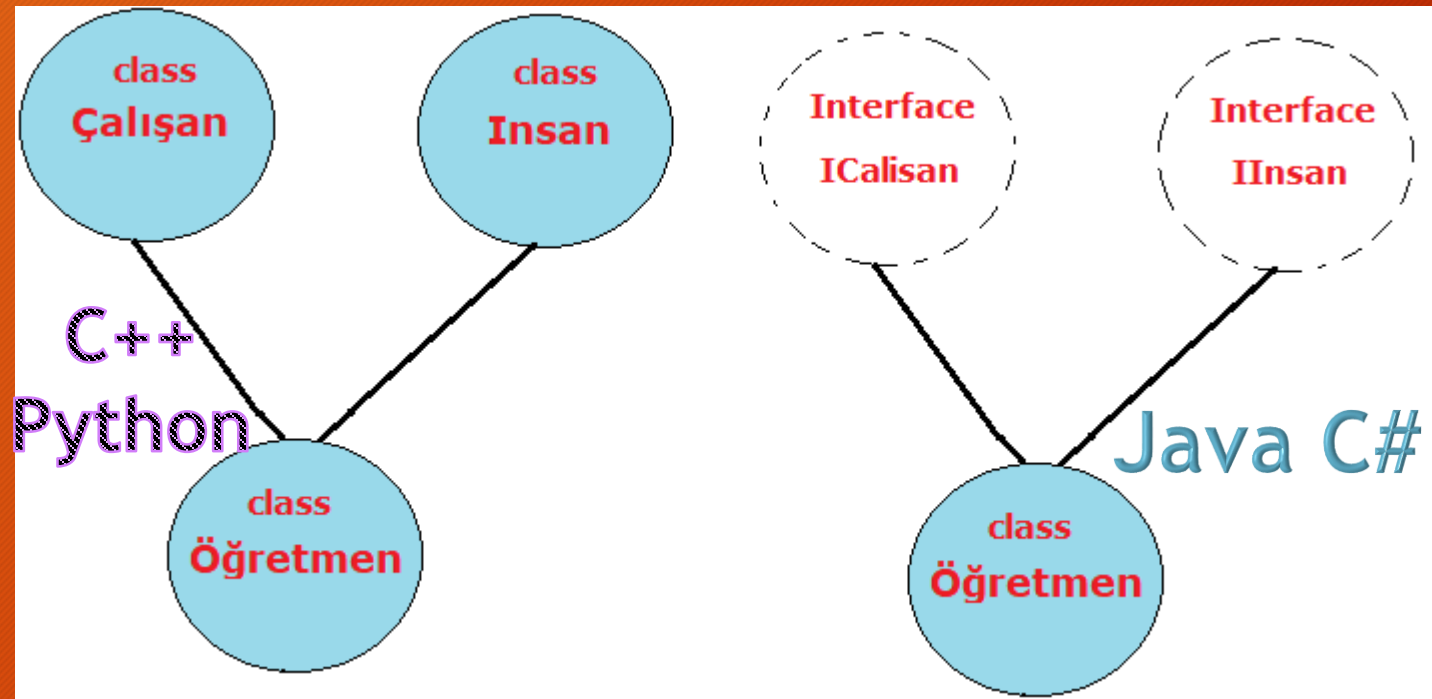
Arayüzler

```
interface IEmail{
    public void setGonderen(String gonderen);
    public void setAlan(String alan);
    public void setMesaj(IIcerik mesaj);
}
interface IIcerik{
    public String StringOlarakIcerik();
}
class Email implements IEmail{
    public void setGonderen(String gonderen){ /*Göndereni ayarlar*/ }
    public void setAlan(String alan){ /*Alanı ayarlar*/ }
    public void setMesaj(IIcerik mesaj){ /*Mail mesajını ayarlar*/ }
}
```

```
class Html implements IIcerik{
    public String StringOlarakIcerik(){
        // HTML'i String olarak ifade et.
    }
}
```


Arayüzler

- Çoklu sınıf kalıtımı izin verilmeyen dillerde çözüm olarak arayüz bulunmalıdır. Çoklu arayüz kalıtımı yapılmalıdır.



Çoklu Sınıf Kalıtımı

```
class Sekil
{
    protected double genislik;
    protected double yukseklik;
    1 başvuru
    public Sekil(double genislik,double yukseklik)
    {
        this.genislik = genislik;
        this.yukseklik = yukseklik;
    }
}
```

C#



base

```
class Daire : Sekil
{
    0 başvuru
    public Daire(double g,double y) : base(g,y)
    {
    }
}
```

```
class İnsan{
    protected:
        double boy;
    public:
        İnsan(double boy) { this->boy = boy; }
};
class Calisan{
    protected:
        int Id;
    public:
        Calisan(int Id){ this->Id = Id; }
};
class Ogretmen : public İnsan, public Calisan{
    public:
        Ogretmen(double boy,int Id) : İnsan(boy),Calisan(Id){
        }
};
```

C++

```
class İnsan:
    ...
class Calisan:
    ...
class Ogretmen(Insan, Calisan):
    super()
```

Python

Yazılan sıraya göre çağırır
Önce İnsan sonra Calisan sınıfın
Kurucusu çağrılır.

namespace (isim uzayları) ve Paketler

- Bir arayüzden farklı olarak isim uzayları kodu organize etmek, derli toplu bir halde bulunmasını sağlar. C++ ve C# dillerinde desteklenir.
- Java dilindeki karşılığı paketlerdir ve klasör mantığı ile saklanır.

```
namespace A{
    double Hesapla() { return 0; }
}
namespace B{
    double Hesapla() { return 1; }
}
int main() {
    cout<<A::Hesapla()<<endl; 0 yazar
    cout<<B::Hesapla()<<endl; 1 yazar
    return 0;
}
```

C++

Soyut Sınıflar

- İçerisinde tamamlanmamış alanlar içeren sınıflara denir.
- Arayüzden farkı tanımlanmış alanlar da içerebilirler.
- Tanımlanmamış alanlar kalıtım yolu ile tanımlanır.
- Eksik tanım içerdikleri için soyut sınıflardan nesne türetilemez.

Soyut Sınıflar

C++

```
class Canli{
    private:
        int yas;
    public:
        virtual void YemekYe () = 0;
        int Yas ()const { return yas; }
};

class Kedi : public Canli{
    public:
        void YemekYe () {
        }
};
```

C#

```
public abstract class Canli{
    public int yas { get; }
    public abstract void YemekYe();
}

public class Kedi : Canli
{
    public override void YemekYe()
    {
    }
}
```

Kalıtım Hiyerarşisi

- Java ve C#



Object Veri Türü

- Herhangi bir türü içinde barındırabilecek şekilde tasarlanmış veri türüdür.
- Boxing ve Unboxing kullanımlarında büyük önem arz eder.
- Java dilinde en üstteki sınıf Object sınıfıdır ve bu sınıftan diğer bütün sınıflar gizli olarak kalıtım alır.
- Java ve C# dillerinde Object sınıfı bulunurken, C++ dilinde böyle bir sınıf yoktur.

Generic - Şablon Yapılar

- C# ve C++ dillerinde aktif olarak kullanılan şablon yapılar Java diline sonradan dahil olmuştur.
- Bir dilde Object sınıfı ile şablon yapıları taklit edilebilir.

```
public class Koleksiyon<Tur>
{
    public Tur[] Elemanlar;

    public Koleksiyon(Tur []Elemanlar)
    {
        this.Elemanlar = Elemanlar;
    }

    public String EnBuyukEnKucukBirlestir()
    {
        return Elemanlar.Max().ToString() + Elemanlar.Min().ToString();
    }
}
```

C#

```
int[] tamsayilar = { 15,95,20,2,18,32};
Koleksiyon<int> koleksiyonsayilar = new Koleksiyon<int>(tamsayilar);
koleksiyonsayilar.EnBuyukEnKucukBirlestir(); 952 döner

char[] karakterler = { 'a', 'w', 'r', 't', 'k', 'p' };
Koleksiyon<char> koleksiyonkarakterler = new Koleksiyon<char>(karakterler);
koleksiyonkarakterler.EnBuyukEnKucukBirlestir(); wa döner
```

Generic - Şablon Yapılar


C++

```
template <typename Tur>
class Koleksiyon{
public:
    Tur *Elemanlar;
    int uzunluk;
    Koleksiyon(Tur *Elemanlar,int uzunluk){
        this->Elemanlar = Elemanlar;
        this->uzunluk = uzunluk;
    }
    string EnBuyukEnKucukBirlestir(){
        return toString(*max_element(Elemanlar,Elemalar+uzunluk))+
            toString(*min_element(Elemanlar,Elemalar+uzunluk));
    }
    string toString(Tur t)
    {
        ostringstream ss;
        ss << t;
        return ss.str();
    }
};
```


friend Erişim Niteleyicisi

```
class Sayi{
    private:
        double deger;
        void setDeger(double dgr){
            deger = dgr;
        }
    friend class Top;
};

class Top{
    private:
        Sayi *s;
    public:
        Top() {
            s = new Sayi();
            s->setDeger(100);
        }
        double getDeger() const{
            return s->deger;
        }
};
```



- Java dilinde **friend** diye bir terim yoktur. Fakat bu işlem, Sınıfları aynı pakete yerleştirerek kısmen gerçekleştirilebilir.
- Aynı paketteki sınıflar birbirlerinin protected ve erişim niteleyicisi olmayan elemanlarına erişebilirler.

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016

Programlama Dillerinin Prensipleri

Hafta 12 - Eş Zamanlı Programlama

Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

- Tanım
- Öncelik Grafları
- Eşzamanlık Şartları
- Fork ve Join Yapıları
- Parbegin ve Parend Yapıları
- Programlama Dillerinde Eş Zamanlılığın Gerçekleştirimi
- İşlem Durumları
- İşlem Grafları
- Kritik Bölge
- Semaforlar

Tanım

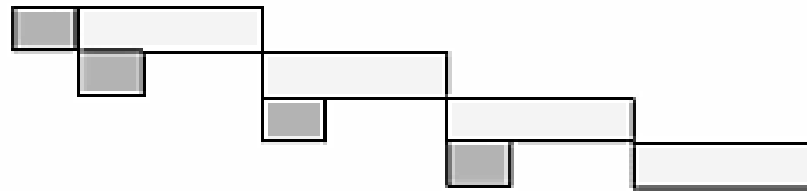
- Programlama dillerindeki eş zamanlılık kavramı ile bilgisayar donanımındaki paralel çalışma birbirinden bağımsız kavramlardır.
- Programlama dillerinde eş zamanlılık belli kavramlar kullanılarak gerçekleştirilebilir.
- Amaç programın hesaplama hızını arttırıp verimi yükseltmektir.
- Bir programda eş zamanlılık kullanılmamışsa varsayılan olarak bir thread o programı yürütecektir.

Donanımsal Paralellik

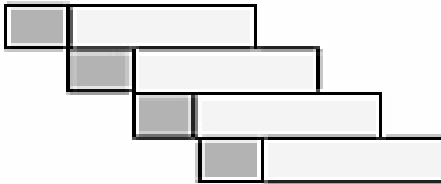
Thread Yok



**Çok Thread
Tek İşlemci**

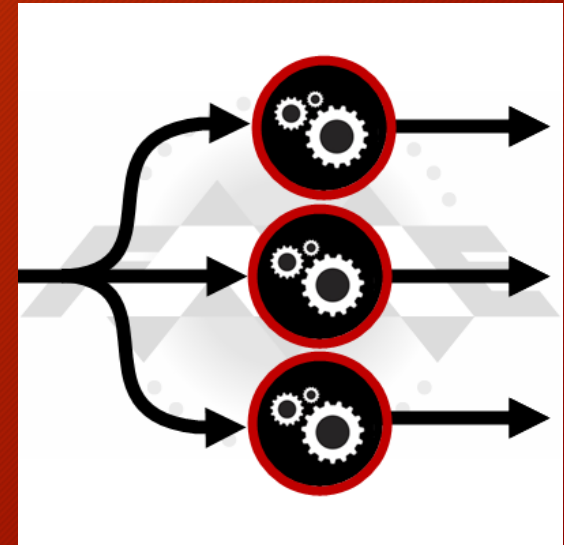


**Çok Thread
Çok İşlemci**



Eş Zamanlılık Türleri

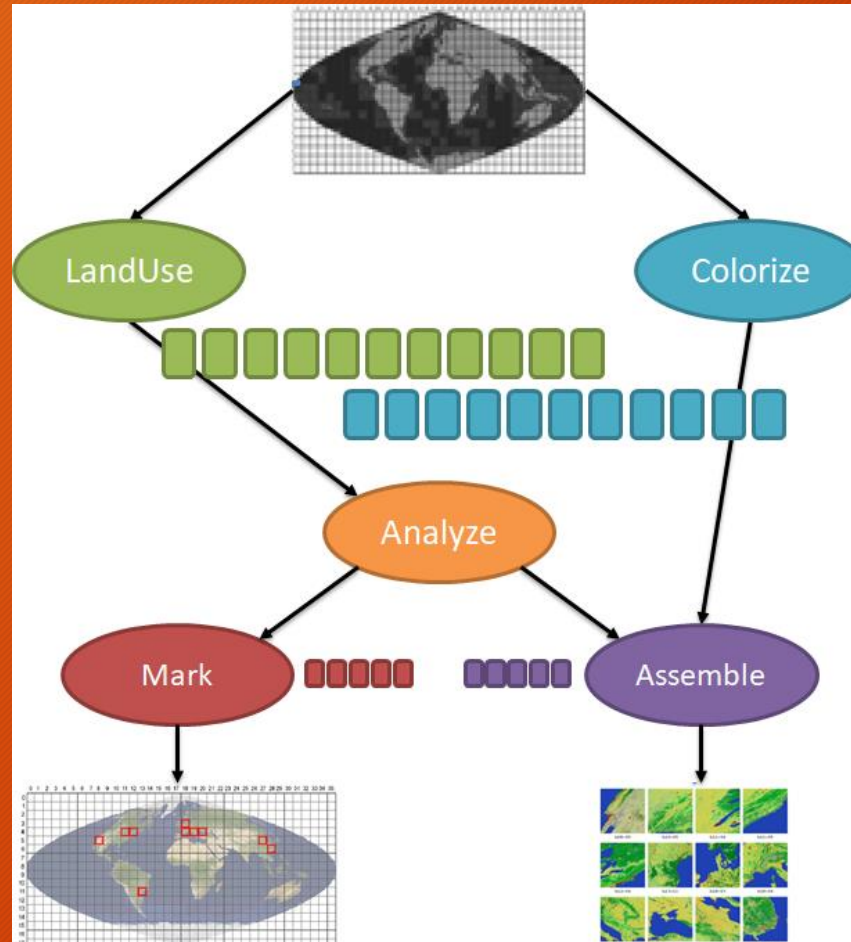
- Program içerisindeki eş zamanlılık 4 farklı şekilde gerçekleşebilir.
 - **Makine komutu düzeyinde** : 2 veya daha fazla makine komutunun paralel çalıştırılması
 - **Program kod satırı düzeyinde**: 2 veya daha fazla program kod satırının paralel çalıştırılması
 - **Birim seviyesinde** : 2 veya daha fazla fonksiyonun paralel çalıştırılması
 - **Program seviyesinde** : 2 veya daha fazla programın paralel çalıştırılması



Programlama Dillerinde Gerçekleştirimi

- Bazı dillerde kütüphaneler yardımıyla gerçekleştirilir.
 - Örnek: OpenMP - C/C++ ve Fortran
- Diğer bazı diller bunu kendi içerisinde sağlayabilir.
- Bu şekilde ilk destek PL/I programlama dili ile başlamıştır.
- Daha sonra bunu, Ada95, Java, C#, Python ve Ruby takip etmiştir.

Python Paralleleştirme Örneği

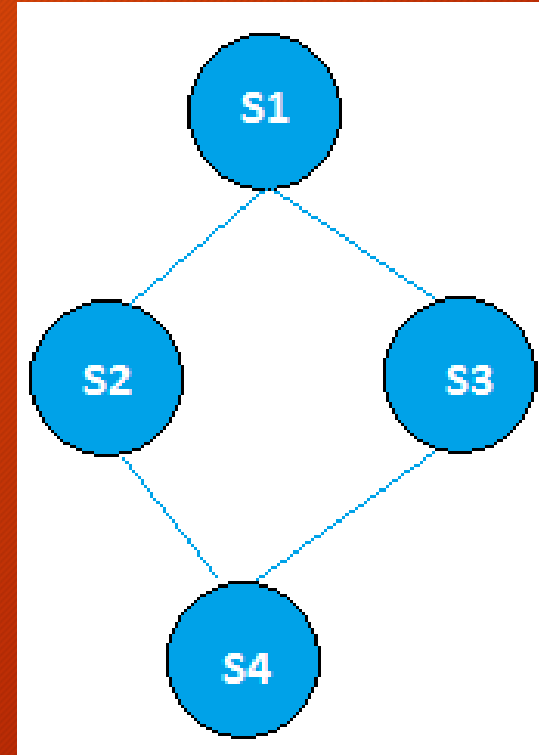


Öncelik Grafları

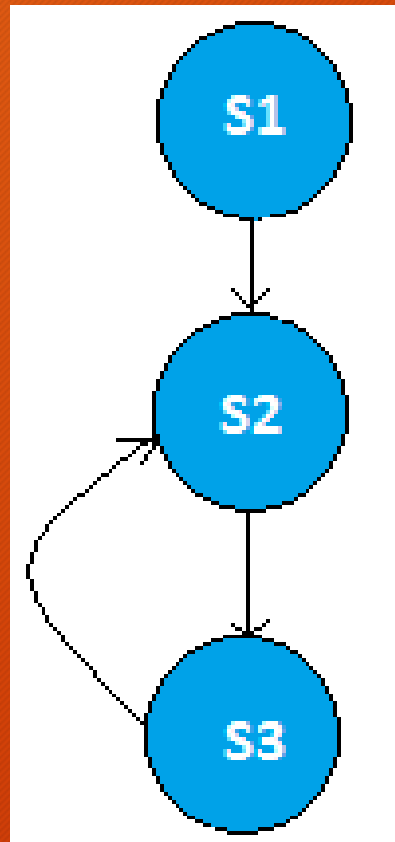
- Bağımlı ve bağımsız değişkenlerin tespit edilip hangi işlerin eş zamanlı çalıştırılabileceği belirlenir.
- Bir öncelik grafi hangi işlemlerin eş zamanlı hangi işlemlerin seri olarak çalıştırılması gerekeceğini gösterir.
- Hangi işlemlerin hangi işlemleri bekleyeceği görülebilir.

Öncelik Grafları

- S2 ve S3 çalıştırılabilmesi için S1 ifadesi işlemini bitirmelidir.
- S2 ve S3 eş zamanlı çalıştırılabilir.
- S4 çalışabilmesi için hem S2 hem de S3 işlemini bitirmelidir.



Hatalı Öncelik Grafi



Eşzamanlık Şartları

$R(S_i) = \{a_1, a_2, \dots, a_n\}$: S_i için “oku” kümesi.

$W(S_i) = \{b_1, b_2, \dots, b_n\}$: S_i için “yaz” kümesi.

S_1 ve S_2 eşzamanlı çalışması için aşağıdaki 3 şart gerçekleşmelidir.

1. $R(S_1) \cap W(S_2) = \{\}$
2. $W(S_1) \cap R(S_2) = \{\}$
3. $W(S_1) \cap W(S_2) = \{\}$

Eşzamanlık Şartları

S1 $a := x + y;$ $R(S1) = \{x, y\}$
S2 $b := z + 1;$ $R(S2) = \{z\}$
S3 $c := a - b;$ $R(S3) = \{a, b\}$
 $W(S1) = \{a\}$
 $W(S2) = \{b\}$
 $W(S3) = \{c\}$

S1 ve S2 deyimleri eş zamanlı olarak çalışabilir mi?

Koşul 1. $R(S1) \cap W(S2) = \{x, y\} \cap \{b\} = \{\}$

Koşul 2. $W(S1) \cap R(S2) = \{a\} \cap \{z\} = \{\}$

Koşul 3. $W(S1) \cap W(S2) = \{a\} \cap \{b\} = \{\}$



S1 ve S3 deyimleri eş zamanlı olarak çalışabilir mi?

Koşul 1. $R(S1) \cap W(S3) = \{x, y\} \cap \{c\} = \{\}$

Koşul 2. $W(S1) \cap R(S3) = \{a\} \cap \{a, b\} = \{a\}$

Koşul 3. $W(S1) \cap W(S3) = \{a\} \cap \{c\} = \{\}$



S2 ve S3 deyimleri eş zamanlı olarak çalışabilir mi?

Koşul 1. $R(S2) \cap W(S3) = \{z\} \cap \{c\} = \{\}$

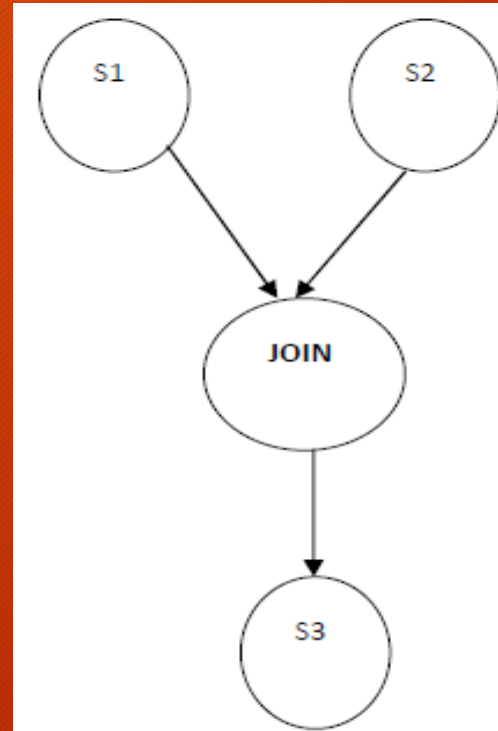
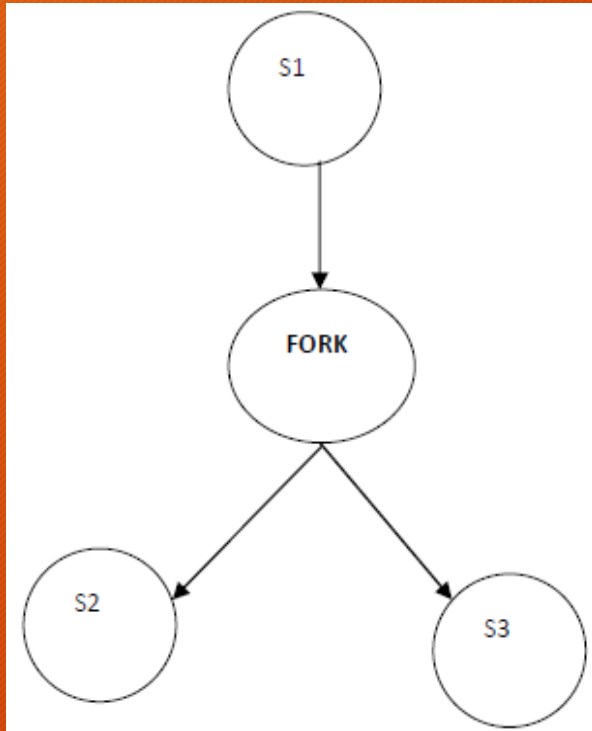
Koşul 2. $W(S2) \cap R(S3) = \{b\} \cap \{a, b\} = \{b\}$

Koşul 3. $W(S2) \cap W(S3) = \{b\} \cap \{c\} = \{\}$



Fork ve Join Yapıları

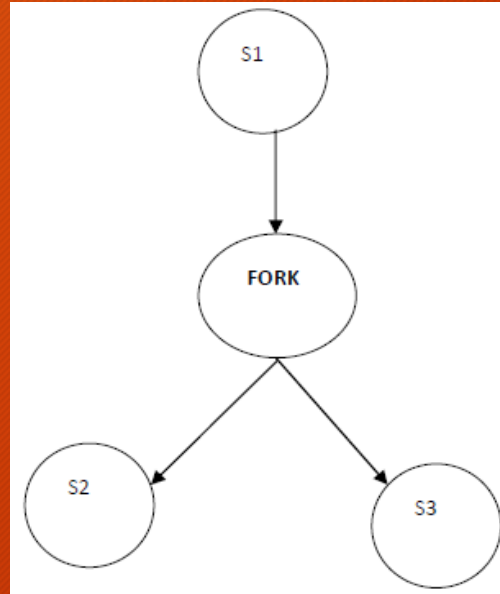
- Eş zamanlılığı tanımlayan ilk programlama dili notasyonlarından biridir.



Fork Yapısı

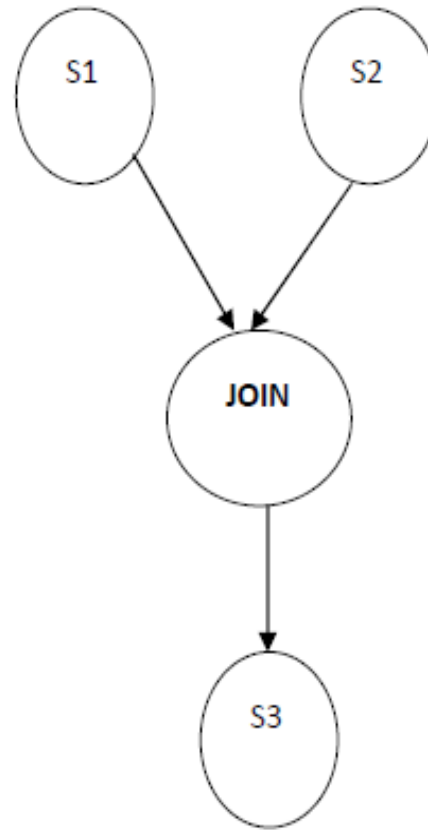
- İki eş zamanlı eşleme üretir.

```
S1;  
FORK L  
S2;  
...  
...  
L:S3;
```



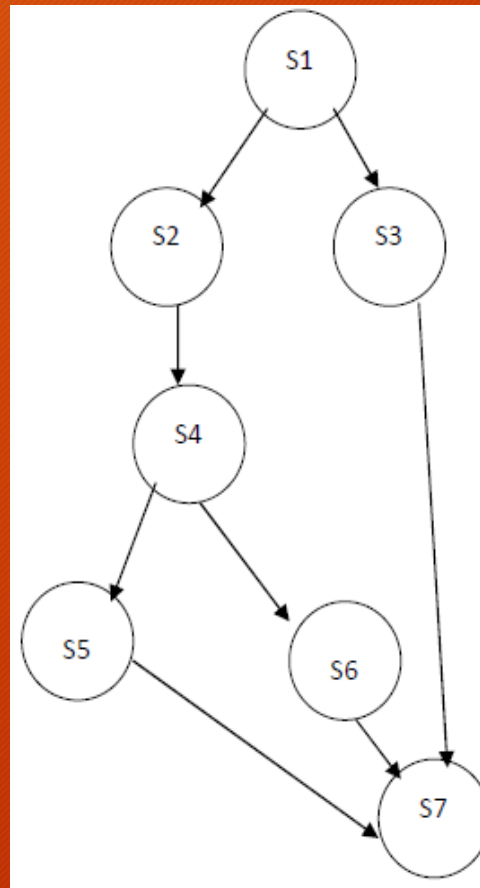
Join Yapısı

```
Count:=2;  
FORK L1;  
...  
...  
S1;  
Go to L2;  
  
L1:S2;  
L2:JOIN count;  
S3;
```



Fork Join Örneği

```
S1  
Count:=3;  
FORK L1;  
S2;  
S4;  
FORK L2;  
S5;  
Goto L3;  
L2:S6;  
Goto L3;  
L1:S3;  
L3: JOIN count;  
S7;
```



Parbegin ve Parend Yapıları

Parbegin

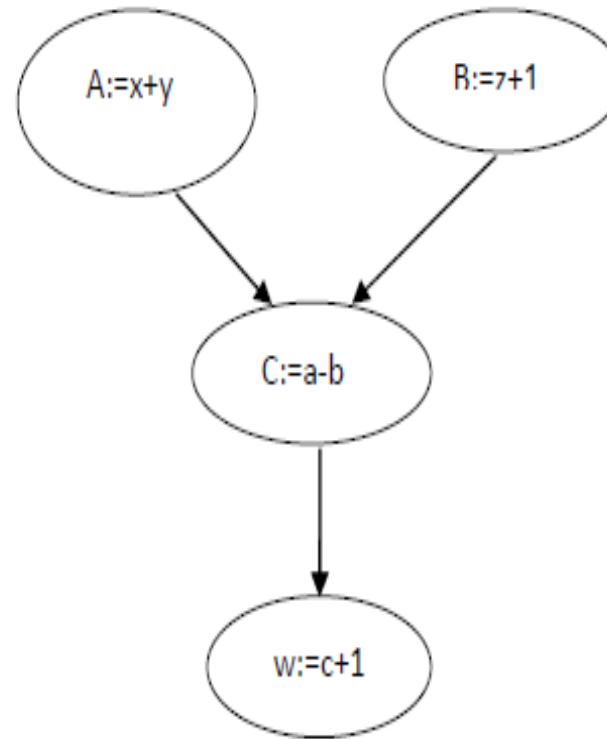
a:= x + y;

b:= z + 1;

parend;

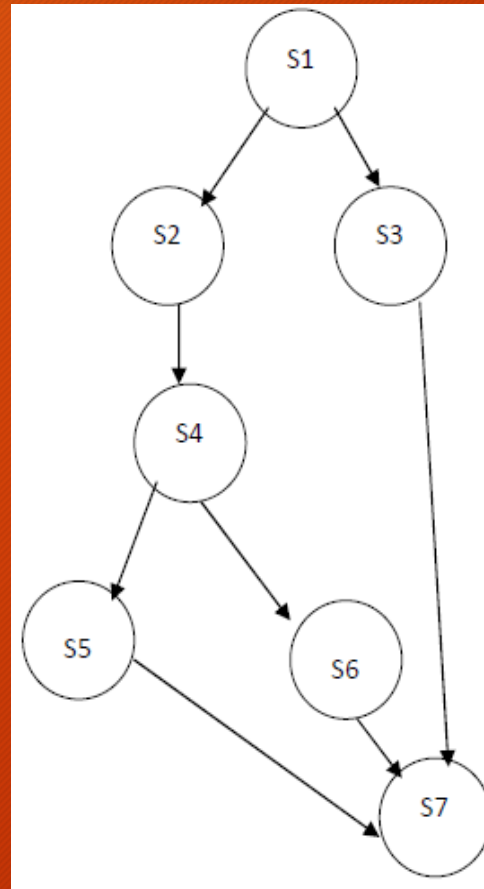
c:= a - b;

w:=c + 1;

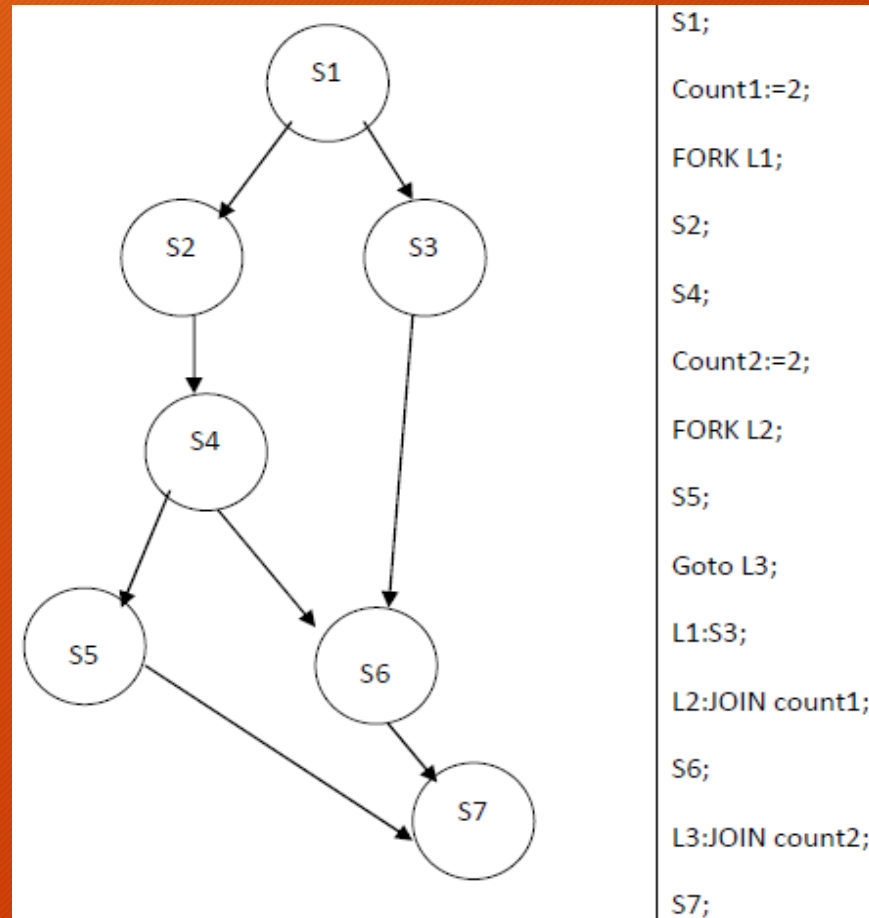


Parbegin ve Parend Yapıları

```
S1;  
Parbegin  
S3;  
Begin  
S2;  
S4;  
Parbegin  
S5;  
S6;  
Parend;  
End;  
Parend;  
S7;
```



Fork Join ile Yapılıp Parbegin Parend ile Yapılamayan Örnek



Fonksiyonel Dillerde Eş Zamanlılık

- Multi-LISP

- 1985 Yılında tanıtılan bu dil program parçalarının eş zamanlı çalışmasına izin veriyordu.
- pcall yapısı ile bu gerçekleştiriliyordu.

(fonk x y z)



Eş Zamanlılık olmayan normal bir fonksiyon çağırımı

(pcall fonk x y z)



Eş Zamanlılık içeren fonksiyon çağırımı

- pcall ile çağırılması x, y ve z parametrelerinin eş zamanlı çalışmasını sağlayacaktır. x y ve z parametreleri yine bir fonksiyon olabilir bu durumda bu fonksiyonlar eş zamanlı çalıştırılır.

Thread Yield Metodu

- yield metodu geçici olarak diğer threadlere zaman verir.
- `Thread.yield();` şeklinde kullanılır.
- O satıra gelen her thread bu komutu uygulayacaktır.

Thread Sleep Metodu

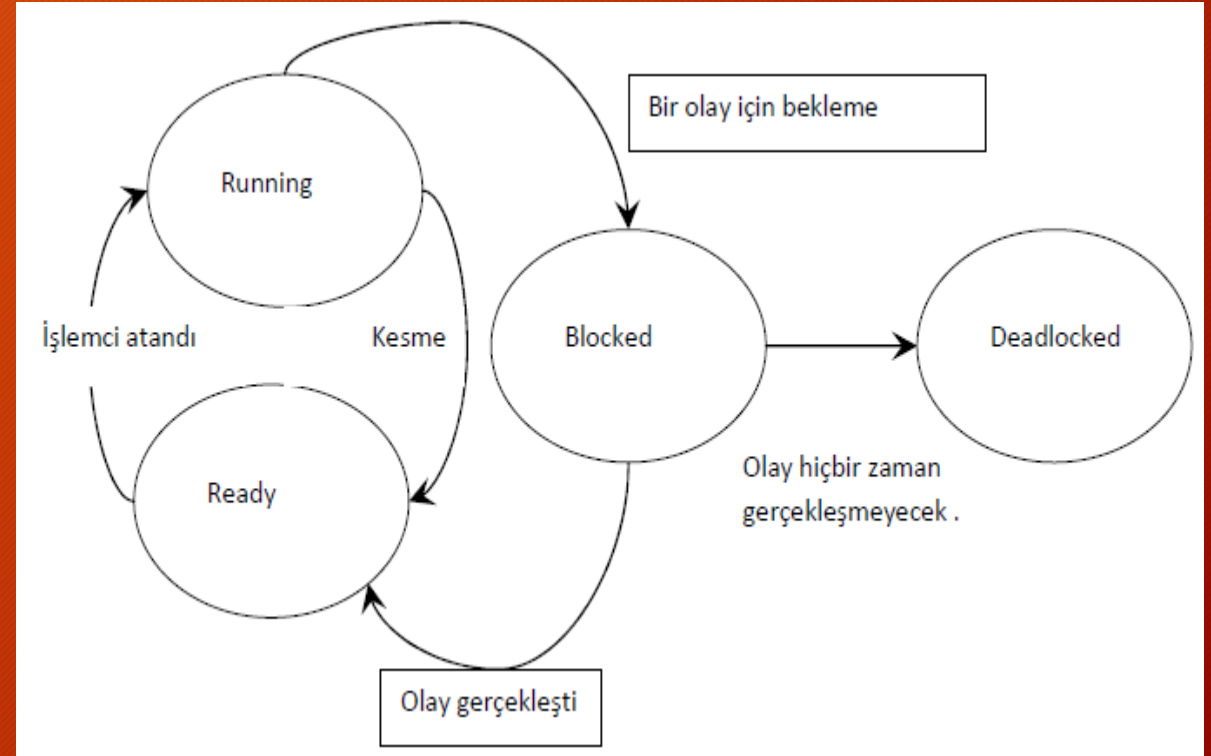
- Belirtilen süre boyunca thread uykuya geçer.
- Bu tepki süresinin uzun olduğu bazı durumlarda zorunlu olarak kullanılabilmektedir.
- `Thread.sleep(1);` 1 milisaniye uykuya geçirir.
- Java dilinde sleep metodu yakalanması zorunlu bir hata fırlatma durumu olduğu için try catch bloklarında kullanımı zorunludur.

Sleep ve Yield Metotları

- Bu metotların kullanımı için eş zamanlılık şart değildir.
- Programa atanan varsayılan thread bu komutlara denk geldiğinde işleyişi uygular.

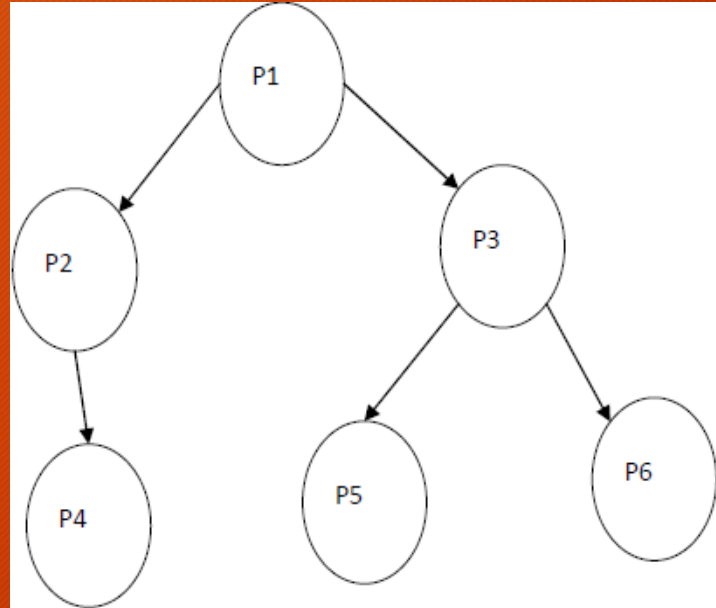
İşlem Durumları

- Running: Komutlar işletiliyor.
- Blocked: Sistem bazı durumlar için bekletiliyor.
- Ready: İşlem bir işlemciye atanmak için hazır durumda bekletiliyor.
- Deadlock: İşlem hiç bir zaman gerçekleşmeyecek olayları bekliyor.



İşlem Grafları

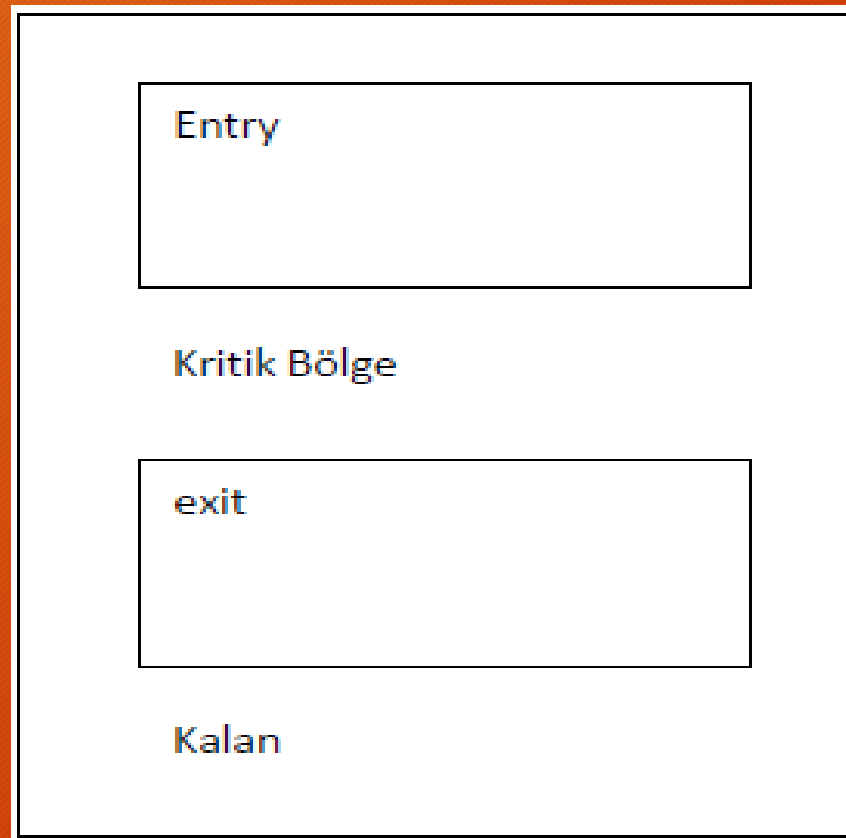
- Bir işlem grafi düğümleri işlemlere karşılık gelen yönlendirilmiş köklü bir graftır.
- P_i düğümünden P_j düğümüne gelen ok işareti P_i 'nin P_j 'yi oluşturduğunu ifade eder.



Kritik Bölge

- Birlikte çalışan n adet işlemden $\{P1, P2, \dots, Pn\}$ oluşan bir sistem olduğu düşünülürse.
- Her bir işlem ortak değişkenleri okuyan bir tabloyu güncelleyen, bir dosyayı yazan vb. işlemleri içerebilir.
- Bu bölümlere kritik bölge ismi verilir.
- Bu bölgelere aynı anda sadece bir thread girmelidir.

Kritik Bölgenin Yapısı



Kritik Bölge Gerçekleştirimi için Yaklaşımlar

Örnek Algoritma 1

Repeat

While turn \neq i do skip;

Kritik Bölge

Turn=j;

Kalan

Until false;

Analiz:

- Algoritma hangi işlemin kritik bölgesine girmesine izin verdiğini hatırlar
- İşlemin hangi aşamada olduğunu hatırlayamaz.

Kritik Bölge Gerçekleştirimi için Yaklaşımlar

Örnek Algoritma 2

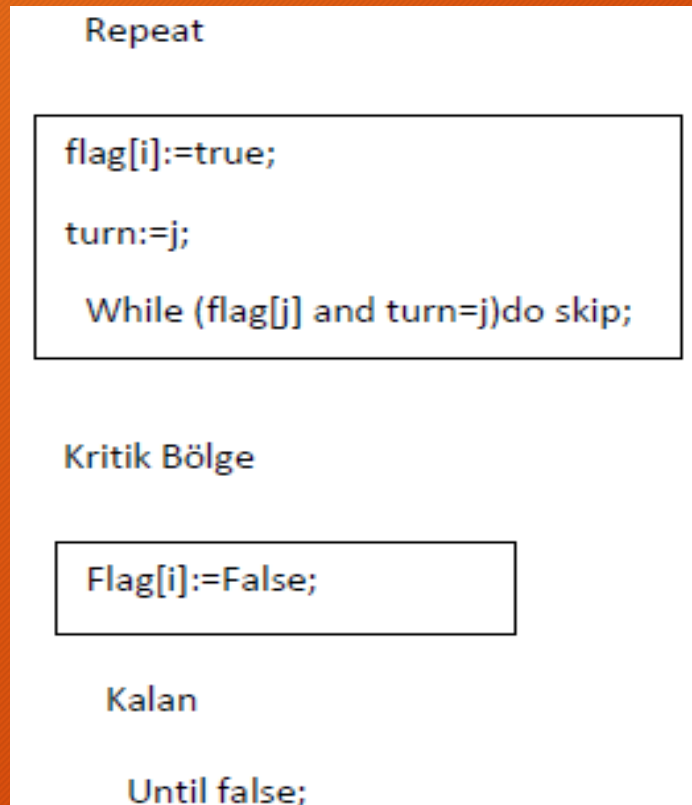


Analiz:

- Algoritma hangi işlemin kritik bölgesine girmesine izin verdiğini hatırlar
- İşlemin hangi aşamada olduğunu hatırlar.
- Fakat bir dizi kontrol edildiği için aynı anda birden fazla thread'in kritik bölgeye girme ihtimali vardır.

Kritik Bölge Gerçekleştirimi için Yaklaşımlar

Örnek Algoritma 3



Analiz:

- Algoritma hangi işlemin kritik bölgesine girmesine izin verdiğini hatırlar
- İşlemin hangi aşamada olduğunu hatırlar.
- Birden fazla thread'in kritik bölgeye girmesine ihtimal yoktur.

Semaforlar

- Karşılıklı hariç bırakma (mutual exclusuion) problemi için yapılan çözümleri daha kompleks problemler için genelleştirmek kolay değildir.
- Bu zorluğun üstesinden gelebilmek için semaforlar olarak adlandırılan bir senkronizasyon aracı kullanılabilir.

Semaforlar

Repeat

P(Mutex)

Kritik Bölge

V(Mutex)

Kalan

Until false;

S1;

V(synch);

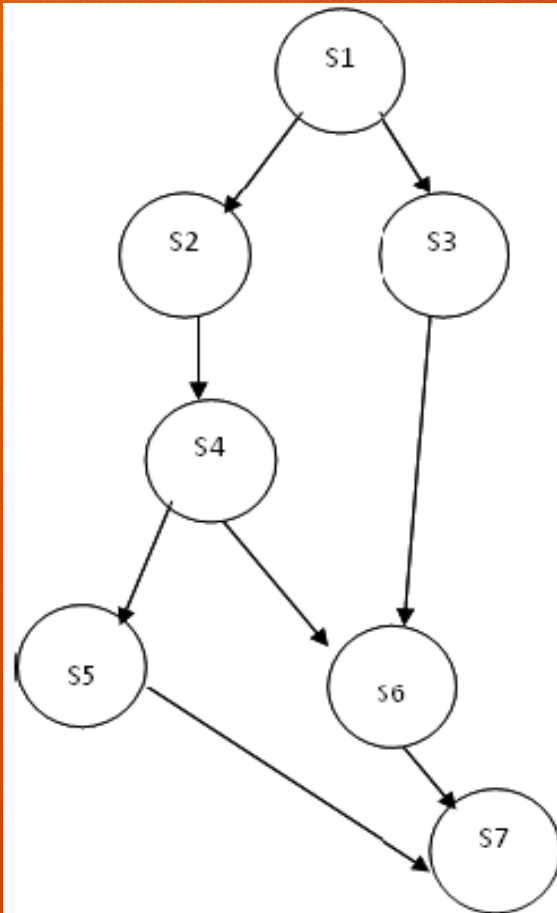
P(synch);

S2;

P1 işlemine konulur.

P2 işlemine konulur.

Semaforların Örnek Üzerinde Kullanımı



```
Var a, b, c, d, e, f, g: semaphore;  
  
Begin  
  Parbegin  
    Begin S1; V(a); V(b) end;  
    Begin P(a); S2; S4; V(c); V(d); end;  
    Begin P(b); S3; V(e); end;  
    Begin P( c); S5; V(f); end;  
    Begin P(d); P(e); S6; V(g); end;  
    Begin P(f); P(g); S7; end  
  Parend;  
End;
```

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016

Programlama Dillerinin Prensipleri

Hafta 14 - Fonksiyonel Programlama

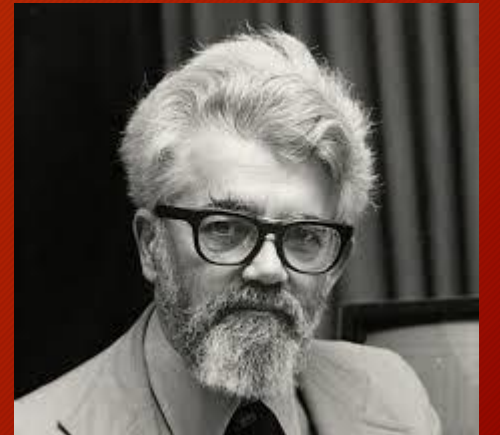
Dr. Öğr. Üyesi M. Fatih ADAK

İçerik

- Tarihsel gelişim
- Tanım
- Neden fonksiyonel paradigma?
- Yaklaşım
- Soy ağacı
- Fonksiyonel dillerin yapısı
- Değişkenin rolü
- Haskell dili
- Lisp dili
 - Formlar
 - Veri türleri
- Fonksiyonel ile Emir Esaslı karşılaştırılması

Tarihsel Gelişim

- Fonksiyonel tasarım ilk John McCarty tarafından 1956 yılında tanıtılmıştır.
- En güçlü temsilcisi Lisp dilidir. Bu isim güçlü liste işlemleri yapabilmesinden gelir.



John McCarthy (1927 - 2011)

Tanım

- Fonksiyonel dillerin tasarımı Matematiksel Fonksiyonlara dayalıdır ve değişkenler(variables), matematikte olduğu gibi gerekli değildir.
- Kullanıcıya yakın olan sağlam bir teorik temele sahiptir.
- Fonksiyonel programlamada , bir fonksiyon aynı parametreler verildiğinde daima aynı sonucu üretir (referential transparency).

Neden Fonksiyonel Paradigma?

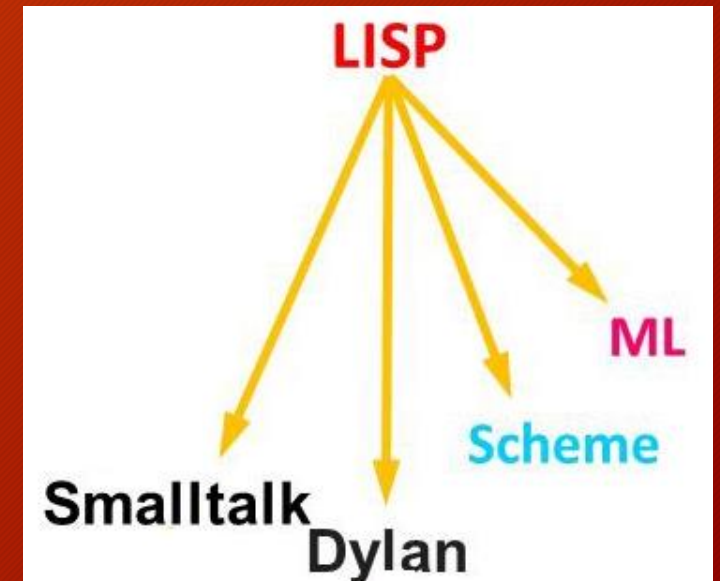
- Tarihsel süreçte emir esaslı tasarımdan sonra tanıtılmıştır.
- Emir esaslı dillerin tasarımı doğrudan doğruya von Neumann mimarisine dayanır.
- Bir emir esaslı dilde, işlemler yapılır ve sonuçlar daha sonra kullanım için değişkenlerde(variables) tutulur. Emir esaslı dillerde değişkenlerin yönetimi karmaşıklığa yol açar.

Yaklaşım

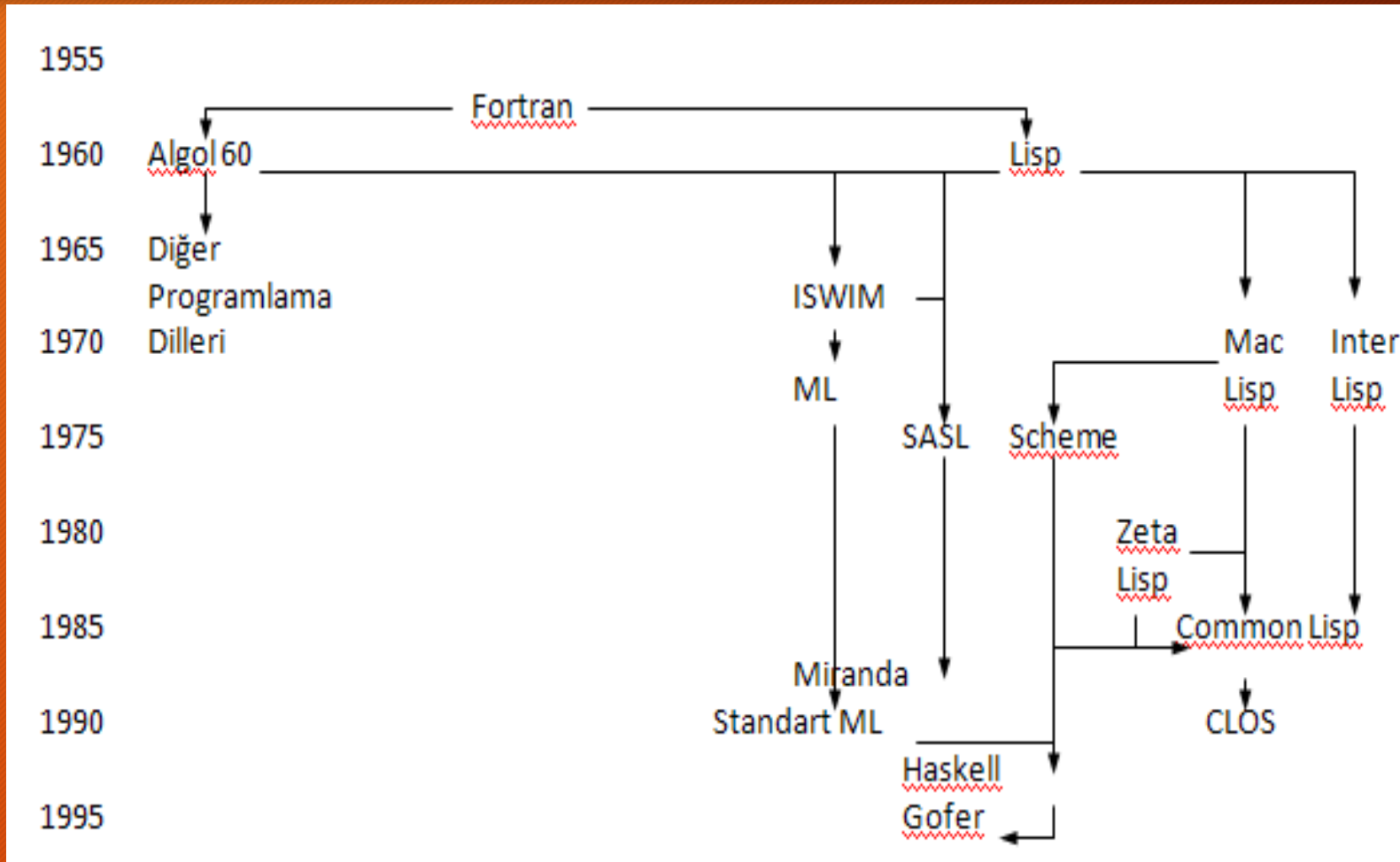
- Fonksiyonel dillerde problemin nasıl çözüleceğinden çok problemin ne olduğu önemlidir.
- For, if, while gibi denetim mekanizmaları makrolar halinde sunulur ve özyineleme ile gerçekleştirilir. Daha çok yapay zeka ve benzetim uygulamaları için uygun olabilir.
- Fonksiyon yaklaşımından dolayı matematik temeli oldukça sağlam olacağından optimize edilme (en iyileme) şansı çok yüksektir.

Yaklaşım devam...

- Fonksiyonel programlama paradigması, Programlama dilini fonksiyon tanımının temel biçimleri üzerine oturarak, algoritmaların ifadesi için basit ve açık bir ortam elde etmeyi amaçlamıştır.
- İlk örnek LISP dilidir ve onu ML, Scheme ve Haskell, dilleri izlemiştir.



Soy Ağacı



Fonksiyonel Dillerin Yapısı

- Sadece fonksiyonlar üzerine kurulmuş bir modeldir.
- Fonksiyonlar bir çok değer alır ve geriye sadece bir değer döndürürler.
- Fonksiyonlar başka fonksiyonları çağırır ya da başka fonksiyonun parametresi olurlar.
`Fonksiyon(..(fonksiyon2(fonksiyon1(veriler)))..)`
- Bu dillerde, alt yordamlar, fonksiyonlar (prosedürler) kullanılarak program daha alt parçalara bölünür.

Fonksiyonel Dillerin Yapısı devam...

- Fonksiyonel diller Sembolik veri işleme amacı ile dizayn edilmiştir.
- Bu diller;
 - Türev ve integral hesaplamalarındaki
 - Elektrik devre teorisindeki
 - Matematiksel mantık oyunlarındaki
 - Yapay zekanın diğer alanlarındaki

sembolik hesaplamalarda kullanılmaktadır.

- Karmaşık hesaplamalar daha basit ifadeler cinsinden yazılarak kolaylıkla çözümlenebilir.

Değişkenin Rolü

- Fonksiyonel olmayan tasarımlarda değişken, bir değeri tutan yer rolünü üstlenirken fonksiyonel tasarımda direk değerin kendisidir.

$x = x + 1$ ifadesinde her x farklı bir değeri temsil eder.

$10 = 9 + 1$ deki gibi düşünülebilir.

Haskell Dili

- Bağımlı ve bağımsız değişkenlerin tespit edilip hangi işlerin eş zamanlı çalıştırılabileceği belirlenir.
- Tam olarak fonksiyonel bir dildir. (değişkenler yoktur, atama ifadeleri yoktur, hiçbir çeşit yan etki yoktur).
- Tembel değerlendirme(lazy evaluation) kullanır (değer gerekmediği sürece hiçbir alt-ifadeyi değerlendirme)
- Liste kapsamaları(list comprehensions), sonsuz listelerle çalışabilmeye izin verir.

Lisp Dili Formları

- ANSI Common Lisp (cLisp)
 - Derleyici, yorumlayıcı, debugger içerir
- GNU Common Lisp (gcl)
 - Derleyici, yorumlayıcı içerir
- Allegro CL (Commercial Common Lisp Implementation)

Lisp Dili Veri Türleri

- İki ana veri türünden oluşur.
 - Atom ve List
- Atom Veri Türü
 - String
 - Tam ve Ondalık sayılar
 - Karmaşık sayılar

Fonksiyonel ile Emir Esaslı Tasarım Karşılaştırması

Emir Esaslı (imperative) diller	Fonksiyonel diller
Verimli çalışma	Verimsiz çalışma
Karmaşık semantik	Basit semantik
Karmaşık sentaks	Basit sentaks
Eş Zamanlılık (kullanıcı tanımlı)	Eş Zamanlılık (Otomatik)

Kaynaklar

- Yumusak N., Adak M.F. *Programlama Dillerinin Prensipleri*. 1. Baskı, Seçkin Yayıncılık, 2018
- Sebesta, Robert W. *Concepts of programming languages*. 11 ed. Pearson Education Limited, 2016.
- Sethi, Ravi. *Programming languages: concepts and constructs*. Addison Wesley Longman Publishing Co., Inc., 1996.
- Watt, David A. *Programming language design concepts*. John Wiley & Sons, 2004.
- Malik, D. S., and Robert Burton. *Java programming: guided learning with early objects*. Course Technology Press, 2008.
- Waite, Mitchell, Stephen Prata, and Donald Martin. *C primer plus*. Sams, 1987.
- Hennessey, Wade L. *Common Lisp*. McGraw-Hill, Inc., 1989.
- Liang, Y. Daniel. *Introduction to Java programming: brief version*. pearson prentice hall, 2009.
- Yumusak N., Adak M.F. *C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar*. 2. Baskı, Seçkin Yayıncılık, 2016