“Bilgisayar Bilimi”

Dr. Cahit Karakuş, 2021

Her 21. yüzyıl öğrencisi, algoritmalar, uygulamaların nasıl yapıldığı ve internetin nasıl çalıştığı hakkında bilgi edinme şansına sahip olmalıdır.

Bilgisayar bilimcileri, Bilgisayar Bilimini zenginleştirip dönüştürürken, doğal kaynaklarımızı yönetme ve tahsis etme şeklimizin verimliliğini ve etkinliğini artırmada kilit bir rol oynayabilir ve oynamalıdır.

Technology affects every field

- Gerçek şu ki teknoloji ticaretin her alanını etkiliyor
- Sağlık hizmetlerinde - bilgisayar her gün ameliyathanelerin bir parçasıdır ve diyabetli insanlar için insülin seviyelerini tespit eden bu kontakt lensler gibi buluşları mümkün kılmaktadır.
- Uzayda - insanların şimdi yapamayacağı yerleri keşfetmek için bir robot nesline güveniyoruz.
- Evlerimizde -- ısıtma sistemlerimiz gibi günlük şeyleri otomatikleştiriyoruz
- Yollarımızda - bizi eve götürmek için navigasyon sistemlerine güveniyoruz ve şimdi kendi kendine giden arabaları günlük hayatımıza sokmayı deniyoruz
- Eğlencede – gişe rekorları kıran filmler, yeni karakterlere hayat vermek ve bize tamamen animasyonlu yeni dünyalar sağlamak için bilgisayar bilimine bağlıdır.Ve her gün bu trend her sektörde büyüyor

Bilgisayar Mühendisliği

- Bilgisayar Bilimi (Computer Science)
- Bilgisayar Mühendisliği (Computer Engineering)
- Yazılım Mühendisliği (Software Engineering)
- Bilişim Sistemleri Mühendisliği (Information Systems Engineering)
- Bilgisayar: Sayısal ya da mantıksal işlem yapabilen programlanabilen makine
- Bilgisayar Bilimleri: Hesaplama olgusunun kuramı ve gelişimi ile ilgilenen temel bilim
- Bilgisayar Mühendisliği: Bilgisayarlı tabanalı sistemlerin işlevsel yazılımlarını geliştirmek amacıyla farklı temel bilimleri kullanma disiplini.

Bilgisayar Mühendisi olmak için en temel ihtiyaç nedir?

- **Bilgisayar bilimi**, bilgisayarların tasarımı ve kullanımı için temel oluşturan hesaplamalara ve uygulamalarına yönelik bilimsel ve pratik bir yaklaşım geliştirilen teori, deney ve mühendislik çalışmasıdır.
- **Bilgisayar bilimi**; edinim, temsil, işleme, depolama, iletişim ve erişimin altında yatan yönteme dayalı prosedürlerin veya algoritmaların fizibilitesi, yapısı, ifadesi ve mekanizasyonunun sistematik çalışmasıdır.

Bilgisayar Bilimi

- Gerçek şu ki, bilgisayar bilimi mantık, problem çözme ve yaratıcılıkla ilgilidir.
- Herhangi bir bağlamda çözmeye çalışılan problemler hakkında nasıl farklı düşüneceklerini öğretir.
- Nasıl dijital eserler yaratılacağını ve bu eserlerin gizlilik ve güvenlik gibi konulara bakarak etraflarındaki dünyayı nasıl etkilediğini öğretir.
- Bunu düşünmenin çok basit bir yolu, CS'nin çocuklara sadece teknoloji tüketicisi olmak yerine nasıl yeni teknolojileri YARATACAKLARINI öğretmesidir.
- Bilgisayar biliminin ne olduğunu anlamak kadar, ne olmadığını anlamak da önemlidir.
- Teknoloji yaratıcıları yapabilmektir.

CS

- CS = the science of computers
  - not a “natural science”, but: a “science of the artificial”
  - “computers” includes: hardware, algorithms, etc.
  - CS = the artificial **science** & engineering of **computers**
  - CS = the “study” of algorithms
    - Algorithms ≈ what you can teach a computer
    - Which functions (I-O) can be efficiently computed?
    - Need a computer to find out!
  - CS = study of algorithms
  - CS = study of information
    - how to represent info
    - how to process info
    - & the machines that do this
  - CS = natural science of procedures
    - including algorithms, …
    - & recipes (specifications; vague)
    - & non-halting (& interactive) procedures
    - & heuristics (incorrect O/P)

CS

- CS ≠ science
  - not concerned with “discovery”
- CS = **engineering**
  - concerned with “making”:
    - physical computers
    - S/W systems
- CS =
  - discovery (science)
  - & implementation (engineering)
  - of information processes

Computer Science

Teorik Bilgisayar Bilimi:

- computation
- Information and Coding
- Data structures and algorithms
- Programming language theory and formal methods

Applied computer science

- Computational science, finance and engineering
- Social computing and human–computer interaction
- Software engineering: Matlab, C++, Python, Java Script, Assemble

Computer systems and computational processes

- Artificial intelligence
- Computer architecture and organization
- Concurrent, parallel and distributed computing
- Computer networks
- Computer security and cryptography
- Databases and data mining
- Computer graphics and visualization
- Image and sound processing

Bilgisayar Bilimi

- Bilgisayar, ***veri işleme yeteneği olan bir yapıdır.***
- Bilgisayar bilimleri, var edilen değerlere sadece fiziksel materyaller ile değil, bilgi işleme ve hesaplama becerileri ile anlam kazandıran mühendislik disiplin dalıdır.
  - Değer: mal, hizmet, fikir ve kültürel ürünlerdir.
  - Teorik ya da pratik, büyük ya da küçük, basit ya da karmaşık problemlere bilgisayar sistemleri ile donamımsal ve yazılımsal çözüm sağlayacak verimli ve doğru yöntemleri tasarlama sanatıdır.
- **Bilgisayar Bilimi**, verinin bir amaç doğrultusunda hangi kurallarla ve yordamlarla işlenmesi gerektiğiyle ilgilenen bir bilim dalıdır.
- Bilgisayar bilimleri çalışma alanı sadece “bilgisayar” ile kısıtlı olmayan bir bilim dalıdır. Bilgi işleme ve hesaplama (Computation) ile ilgili herşeyi içerir. Matematiksel modeller geliştirilir. Algoritma tasarlar ve yazılımsal çözümler üretir.
- Temelde **Mantık** ve **Matematik** bilimlerinin üzerine kurulmuştur ve **Elektrik, Elektronik** bilimlerinin teknolojilerini kullanır.

Bilgisayar Bilimi

- Bilgisayar bilimi, bilgisayar tabanlı sistemlerin tasarımı ve kullanımı için temel oluşturan teori, deney ve mühendislik çalışmasıdır. Hesaplamaya ve uygulamalarına bilimsel ve pratik bir yaklaşımdır.
- Bilgisayar bilimi; edinim, temsil, işleme, depolama, iletişim ve erişimin altında yatan yönteme dayalı prosedürlerin veya algoritmaların fizibilitesi, yapısı, ifadesi ve mekanizasyonunun sistematik çalışmasıdır.
- Bilgisayar biliminin alternatif, daha özlü tanımı "büyük, orta veya küçük ölçekli algoritmik işlemleri otomatikleştirme çalışması" olarak nitelendirilebilir.
- Bir bilgisayar bilimcisi, hesaplama teorisi ve hesaplama sistemlerinin tasarımı konusunda uzmanlaşmıştır.
- Bilgisayar grafikleri gibi alanlar, gerçek dünya görsel uygulamalarını vurgularken, hesaplamalı karmaşıklık teorisi (hesaplama ve zor olan sorunların temel özelliklerini araştıran) gibi bazı alanlar oldukça özeldir.
- İnsan-bilgisayar etkileşimi alanı; bilgisayarları ve hesaplamaları yararlı, kullanışlı ve evrensel olarak insanlara ulaştırmaya yönelik zorlukları tanımlamaya ve onları aşmaya çalışmaktadır.

Bilgisayar Bilimi

- Problem çözmenin tüm yönlerini kapsar:
  - Matematiksel modelleme ve algoritma çözümleme ve tasarım
  - Algoritmaların programlar halinde formüle eden programları çalıştıracak, bilgi işleme ve hesaplama cihazların geliştirilmesi
  - Algoritma ve hesaplama modellerinin gücü ve kısıtları ile ilgili daha teorik sorularla da ilgilenir
- Bazıları “bilim” deyince karmaşık bir olayı (phenomena – doğa olayı) anlamayı ve bilimsel yöntem kullanarak problem çözmeyi içermesi gerektiğini söyler.
- Bilimsel yöntem
  - Gözlemlenen davranış ya da olaylar ile ilgili hipotezler oluşturulur.
  - Gözlemlenen davranış ya da olaylar ile ilgili deneyler ve simülasyonlar tasarlar.
  - Uygulamaya yönelik deney alt yapısı kurar ve deneyi uygular.
  - Çözüme yönelik deney sonuçlarını analiz eder. Çözüm hipotezi desteklemezse ise gözden geçirir ve bilimsel yöntemi tekrar eder.
  - Mühendislikte, yeni gelişmeler ve değişen beklentiler doğrultusunda sistem geliştirmede tekrar etmeler sürekli olmak zorundadır.

Bilgisayar Bilimi

- Pratik tekniklerle birlikte veri ve hesaplamanın teorik temellerini de ele almaktadır.

- Bilgisayarbilimi, bilgisayarların tasarımı ve kullanımı için temel oluşturan teori, deney ve mühendislik çalışmasıdır. Hesaplamaya ve uygulamalarına bilimsel ve pratik bir yaklaşımdır.

- Bilgisayar bilimi; edinim, temsil, işleme, depolama, iletişim ve erişimin altında yatan yönteme dayalı prosedürlerin veya algoritmaların fizibilitesi, yapısı, ifadesi ve mekanizasyonunun sistematik çalışmasıdır. Bilgisayar biliminin alternatif, daha özlü tanımı " algoritmik işlemleri otomatikleştirme çalışması" olarak nitelendirilebilir. Bir bilgisayar bilimcisi, hesaplama teorisi ve hesaplama sistemlerinin tasarımı konusunda uzmanlaşmıştır.

- Alanları teorik ve pratik disiplinlere ayrılabilir. Bilgisayar grafikleri gibi alanlar, gerçek dünya görsel uygulamalarını vurgularken, hesaplamalı karmaşıklık teorisi (hesaplama ve zor olan sorunların temel özelliklerini araştıran) gibi bazı alanlar oldukça özeldir. Diğer alanlar sıklıkla hesaplamanın uygulanması konusunda karşılaşılan zorluklara odaklanmaktadır. Örneğin, programlama dili teorisi, hesaplamanın tanımlamasına yönelik çeşitli yaklaşımları ele alırken, bilgisayar programcılığının kendisi de programlama dili ve karmaşık sistemlerin kullanımının çeşitli yönlerini inceler. Bununla beraber insan-bilgisayar etkileşimi alanı; bilgisayarları ve hesaplamaları yararlı, kullanışlı ve evrensel olarak insanlara ulaştırmaya yönelik zorlukları tanımlamaya ve onları aşmaya çalışmaktadır.

- Bilgisayar Bilimi, bilgisayarların ve hesaplama sistemlerinin incelenmesidir. Elektrik ve bilgisayar mühendislerinin aksine, bilgisayar bilimcileri çoğunlukla yazılım ve yazılım sistemleriyle ilgilenir; buna teori, tasarım, geliştirme ve uygulama dahildir.

Bilgisayar Bilimcileri

- Bilgisayar Bilimi alanındaki başlıca çalışma alanları arasında yapay zeka, bilgisayar sistemleri ve ağları, güvenlik, veritabanı sistemleri, insan bilgisayar etkileşimi, görüntü ve grafikler, sayısal analiz, programlama dilleri, yazılım mühendisliği, biyoinformatik ve bilgi işlem teorisi yer almaktadır.

- Nasıl programlanacağını bilmek bilgisayar bilimi çalışması için gerekli olsa da, bu alanın sadece bir unsurudur. Bilgisayar bilimcileri, programları çözmek ve bilgisayar donanımı ve yazılımının performansını incelemek için algoritmalar tasarlar ve analiz eder. Bilgisayar bilimcilerinin karşılaştığı problemler, soyuttan (bilgisayarlarla hangi problemlerin çözülebileceğinin ve bunları çözen algoritmaların karmaşıklığının belirlenmesinden) elle tutulan cihazlarda iyi performans gösteren, kullanımı kolay uygulamalar tasarlamaya ve somut olanlara kadar uzanır. güvenlik önlemlerini destekleyenler.

Bilgisayar Bilimi ana araştırma ve uygulama alanları

- **Sistem/Donanım**
- **Yazılım**
  - Sistem yazılımı – donanım bileşenlerini kontrol eden programlar. Örn. İşletim sistemi
  - Geliştirme yazılımları - diğer programların geliştirilmesinde araç olan programlar
  - Uyglama yazılımları – çeşitli karmaşık görevler için kullanıcılara yardımcı olan programlar; IE, firefox gibi web tarayıcıları, word, wordperfect gibi kelime işlemciler, power point, frame maker gibi dunum programları, Notepad gibi editorler, oyunlar, vb.
- **Teori**

Bilgisayar biliminin alanları

- Bilgisayar bilimi, bir disiplin olarak, algoritmaların teorik çalışmalarından hesaplama ve hesaplama sınırları çalışmalarına, donanım ve yazılım alanlarında bilgisayar sistemlerinin uygulanmasına ilişkin pratik ve teorik olmak üzere bir dizi konuyu kapsar.
- Bilgisayar disiplininde önemli olduğunu düşünülen dört alan şöyle tanımlanmaktadır:
  - Hesaplama teorisi
  - Algoritmalar ve veri yapıları
  - Programlama metodolojisi ve dilleri
  - Bilgisayar elemanları ve mimarisi

Bilgisayar biliminin alanları

- Teorik bilgisayar bilimleri
  - Hesaplama teorisi
  - Bilgi ve kodlama teorisi
  - Algoritmalar ve veri yapıları
  - Programlama dili teorisi
  - Biçimsel yöntemler
- Uygulamalı bilgisayar bilimleri
  - Yapay zekâ
  - Bilgisayar mimarisi ve mühendisliği
  - Bilgisayar performans analizi
  - Bilgisayar grafikleri ve görselleştirme
  - Bilgisayar güvenliği ve kriptografi
  - Hesaplamalı bilim
  - Bilgisayar ağları
  - Eşzamanlı, paralel ve dağıtılmış sistemler
  - Veritabanları
  - Yazılım mühendisliği

Teorik bilgisayar bilimleri

- Teorik bilgisayar bilimleri, matematiksel ve soyut olarak özetlenebilir.
- Hesaplamanın doğasını anlamak ve bu anlayışın bir sonucu olarak daha etkili metodolojiler geliştirmektir.
- Günümüzde matematiksel, mantıksal, standart kavram ve yöntemleri izah eden tüm makaleler, motivasyonlarının kaynağı olarak bilgisayar bilimi uygulamalarını net bir şekilde belirtip belirtmemesi ölçütünde kabul görmektedir.

Hesaplama teorisi

- Bilgisayar biliminin temel sorusu "Ne, verimlilik sağlayacak biçimde otomatikleştirilebilir?"
- Hesaplama teorisi, nelerin hesaplanabileceği ve bunları gerçekleştirmek için ne kadar kaynak harcanacağı gibi temel sorulara cevap vermeye odaklanmıştır.
- Hesaplanabilirlik teorisi hesaplamanın çeşitli teorik modellerinde hangi hesaplama problemlerinin çözülebileceğini inceler.
- Çok sayıdaki hesaplama problemlerini çözmeye yönelik farklı yaklaşımlarla ilişkili zaman ve mekan maliyetlerini inceleyen hesaplama karmaşıklığı teorisi tarafından ele alınmaktadır.

- "Millennium Prize Problems" problemlerinden biri olan Ünlü P = NP? Problemi, hesaplama teorisinde hala açık ve çözülememiş bir sorundur.

- Bilgisayar Bilimi Altyapı ve Ağları: Bilgi işlem aynı zamanda bilgisayarlar arasındaki bağlantılar, ağlardır.
- Bilgisayar Bilimi Adli Bilişim ve Siber Güvenliktir: Suçu çözmek mi? Bizi güvende tutmak mı?
- Güvenli bilgi?
- Bilgisayar Bilimi Tasarımdır: Modeller yapmak mı? Tasarım arabalar, evler, moda, herhangi bir şey?
- Bu, bilgisayar bilimi yoluyla çok şey başaran insanların beceri setidir. Bu beceriler, teknik ekiplerde çalışmak ve bilgisayarlarla bir şeyler yapmanın yeni yollarını icat etmek için gereklidir.
- Bilgi işlem, görselleştirme ve imgelemenin büyük bir parçasıdır. Filmlerde ve video oyunlarında bilgisayar tarafından oluşturulan grafikler gördünüz. Bilimde de bir problemi veya süreci resmetmek faydalıdır.

Bilgisayar Sistemleri Tanımlar

Bilgisayar Sistemlerinin fonksiyonları ve yetenekleri

Günümüzde, askeri, sağlık, kritik alt yapılar, uzay ve beyin alanlarında kullanılan bilgisayar sistemleri bilgiyi işleyen, analiz eden ve kestirim yapan yazılımlar ile birlikte yoğun olarak kullanılmaktadır. Boyut küçülmesi ve yüksek veri işleme hızları ile birlikte her alanda başta robotik sistemler olmak üzere günümüzde bilgisayar sistemleri, uygarlığın özellikle nesnelerin (IoT) bütünleşik bir parçası olmuştur.

Bilgisayar Sistemlerinin Yetenekleri:

- Aritmetik hesaplamaları gerçekleştirir.
  Mantıksal hesaplamalar yapar
- Karşılatırma yapar.
  Verileri saklar. Verileri çok kısa zamanda arayıp bulur.
  Verileri yazılan program doğrultusunda işler.
  Büyük boyutlu problemleri kısa zamanda çözer.

Bilgisayar Sistemleri iki ana unsurdan oluşurlar: Donanım (hardware), bilgisayarların fiziksel kısımlarına donanım denilmektedir. Ekran, klavye, Sabit disk (harddisk), fare, yazıcı, bellek, mikroişlemci, tarayıcı,… Yazılımı (software): donanımı kullanmak için gerekli programlar. İşletim sistemleri ve altında çalışan bütün programlar.

Bilgisayarlar

- Bilgisayarlar, yeni sonuçları hesaplayarak ve sorguları yanıtlayarak bilgileri işler.
- Bilgisayar giriş ve çıkış bilgileri
- Bilgisayarlar genel amaçlıdır çünkü birçok farklı görevi yerine getirebilirler.
- Bilgisayarlar programlanabilir, böylece işlem sıralarını hatırlayabilirler girdi ve çıktıya sahip genel amaçlı, programlanabilir bir bilgi işlemcisi

Gömülü bilgisayarlar ve robotlar

- İçinde tam teşekküllü bilgisayarlar bulunan makineler: Çamaşır makineleri, uçaklar, ATM'ler vb.
- İçinde tam teşekküllü bilgisayarlar bulunan makineler, son derece güvenilir, öngörülebilir bilgisayar programları gerektirir.
- Bilgisayarlar tarafından kontrol edilen tüm fiziksel mekanizmalar robotik cihazlardır. Tanımlama genel amaçlı ve programlanabilir makinelerle sınırlanır. Robotik kol veya araba

Bilgisayarlar bugün ne yapabilir?

- İş üretkenliği yöneticileri
- Kişisel bilgi yöneticileri
- E-tablolar
- Veritabanı yazılımı
- Masaüstü yayıncılık
- Multimedya ansiklopedileri
- Fiziksel dünya simüle edilir
- Bir müzik videosu üretilir

Bilgisayarlar yarın ne yapabilir?

- Hastalıkları teşhis etmek. Bir bilgisayar semptomlara dayalı bir rahatsızlığı tanımlamasına izin veren kurallarda tıbbi bilgileri toplar.
- Yürüyen, konuşan ve öğrenen robotları kontrol eder. Gözler yerine kameraları kullanarak Taksim'den Beylikdüzü'ne bir otobüsü süren bir program oluşturulur.
- Müzik bestelenir ve sanat yaratılır.

Bilgisayarlar sorunları nasıl çözer?

- İnsanlar, sorunları bir bilgisayarın gerçekleştirebileceği küçük işlemlere ayırır.
- Algoritma yazılır.
- Bilgisayar gerektiren bir sorun çözülür.
- Sorunu bir sorun bildiriminde açıkça belirtilir.
- Net talimatlar veren bir algoritma ile sorun çözülür.
- Talimatları gerçekleştirmek için bir bilgi işlem aracısı kullanılır.

Sorunu açıkça belirtmek

- Ne yapılacağı anlatılır, nasıl yapılacağını değil. Kampüsden Çevre Yoluna nasıl gidebilirim?
- Genel problem sınıfları çözülür. Kuledeki A noktasından B noktasına nasıl gidebilirim? y'nin karekökü nedir?

Bir sorun belirtmek

- Açık sorun bildirimi, belirtim olarak adlandırılır. Girdi olarak hangi bilgileri kullanabiliriz? Sorunumuzun çıktısı veya çözümü nasıl görünmelidir?
- Karekök problemi için belirtim
  - Girdi: Pozitif bir sayı $y > 0$
  - Çıktı: $x^2 = y$ olacak şekilde pozitif bir x sayısı
- Spesifikasyonun belirsiz olmadığından emin olunmalıdır.

Algoritma kullanarak problem çözme

- Algoritma - bir görevi gerçekleştirmek için net bir talimat dizisi
  - iyi düzenlenmiş bir dizi
  - iyi tanımlanmış,
  - uygulanabilir operasyonlar
  - gerçekleştirmek için sınırlı zaman alan

Bilgisayar bilimi nedir?

- Bilgisayarların incelenmesi
- Teori, analiz, tasarım, verimlilik, uygulama ve uygulama dahil olmak üzere algoritmik süreçlerin incelenmesi

Bilgisayarlar

- Bilgisayarları nasıl daha çok bizim gibi hale getirebilir.
- Komutlarında belirli koşullar altında hangi eylemin en iyi olduğuna karar vermelerini sağlayan esnekliğe sahip olmak
- Problemlerin makul bir sürede çözülebilecek çözümleri olduğu
- Sorunları çözmenin en etkili yolları
- Bilgiler nasıl daha güvenli hale getirilebilir
- Bilgisayarlar arasındaki iletişimin nasıl geliştirilir?
- Bilgisayar programlarının kalitesi nasıl artırılır?
- Programlama dilleri nasıl geliştirilir?
- İnsanlar ve bilgisayarlar arasındaki etkileşimi nasıl geliştirilir?
- İnsanların bilgiye erişimi nasıl iyileştirilebilir?
- Bilgisayarlar dünyayı grafiksel olarak modelleme yeteneğini nasıl geliştirebilir?

Bilgisayar Sistemleri Tarih

Bilgisayarları kim icat etti?

- Bilgisayar biliminin iki alanda kökleri vardır
  - Mathematics
    - Alan Turing and the Turing machine (1930s)
    - Hesaplamaların elle nasıl yapılacağına dair kağıt ve kalemle geliştirilen teoriler
  - Engineering
    - John von Neumann and the von Neumann machine (1940s)
    - Elektronik devrelerden fiziksel bilgisayarların nasıl oluşturulacağını gösterdi.

Bilgisayarın Matematiksel Kökleri

- Harazmi: Algoritma, Mantıksal düşünme
- Leibniz's Dream (1600s)
  - Matematiksel algoritmalar için herhangi bir problemi tanımlamamıza ve çözmemize izin verecek evrensel bir dil bulabilir miyiz?
    - Tüm akıl yürütmeyi sabit bir temel kurallara indirgeyin
    - Cümleleri manipüle etmek için sabit kurallarla cümlelerin doğruluğunu veya yanlışlığını belirleyin
- George Boole (1800s)
  - Introduces binary notation of calculation
    - Computers use binary system for logic and arithmetic
- Joseph Marie Jacquard (1752 – 1834) Dokuma Makinesi

Teori hakkında daha fazlası

- David Hilbert (1928)
  - Matematik topluluğuna, matematiksel ifadelerin doğruluğunu oluşturmak ve kontrol etmek için yanılmaz, mekanik bir yöntem bulma konusunda çalışmalar yapmıştır. Bir algoritma ile ilgileniyor
- Alonzo Church, Alan Turing ve Kurt Gödel, Hilbert'in meydan okumasına bir çözüm olmadığına dair argümanlar oluşturuyor.
  - Turing, hesaplama argümanı için kavramsal bir bilgisayar oluşturdu.

Turing Makinesi ve Church-Turing Tezi

- Turing Machine
  - Hesaplama için sınırlı sayıda kurala ve sonsuz miktarda “karalama kağıdına” sahip makine
  - Hiç kimse bir Turing makinesinden daha fazlasını yapabilen fiziksel bir bilgisayar tasarlamamıştır.
  - Makine Hilbert'in problemini çözemedi
- Church-Turning Thesis
  - Turing Makinesi, hesaplama sistemleri ile ne demek istediğimizi yakalar
  - Diğer herhangi bir mekanik bilgi işlem aracı kadar güçlüdür

Mühendislik Kökleri

- Hesap makinelerinin ilk adım gelişimi
  - Abaküs - 5000 yıl önce Orta Doğu'da geliştirildi
  - Pascaline - hesaplama için dişli kullanan ilk mekanik hesap makinesi (1642)
  - Charles Babbage'nin Fark Motoru - matematiksel işlevleri hesaplamak için yüzlerce dişli kullanan kavramsal tasarım (1820'ler)

Elektronik devreler

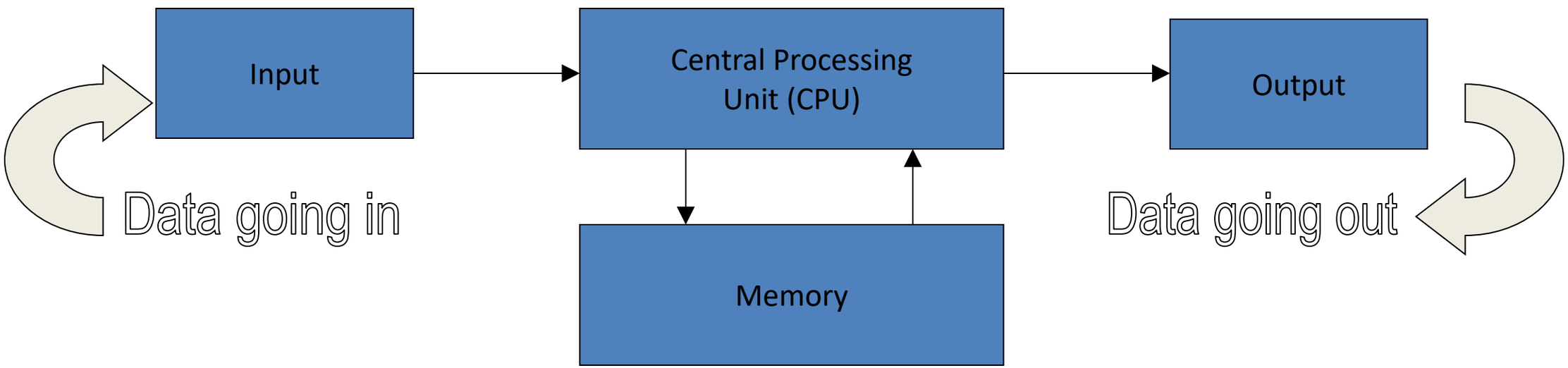
- Telgraf – mesajları iletmek ve bilgileri hızlı bir şekilde iletmek için elektrik kullanır (1844)
- Hollerith Tablolama Makinesi - ABD nüfus sayımını hesaplamak için elektrik ve delikli kartlar kullanır (1890)
- Z2 - aritmetik işlemleri hesaplamak için kullanılan devre (1930'lar)
- Transistor, 1947

Programlanmış Cihazlar

- Joseph Marie Jacquard’ın Tezgah - delikli kartlar kullanılarak belirtilen bir deseni kullanarak kumaş dokur (1801)
- Analitik Motor – Mill, Store, Printer ve Reader'lardan oluşan bir makine için kavramsal tasarım. Alt program gibi programlama kavramlarını tanımlamak için Ada Lovelace'i yönetti
- ENIAC - ilk programlanabilir elektronik bilgisayarlardan biri (1945). Yönlendirme kabloları ve çevirmeli anahtarlarla programlanır

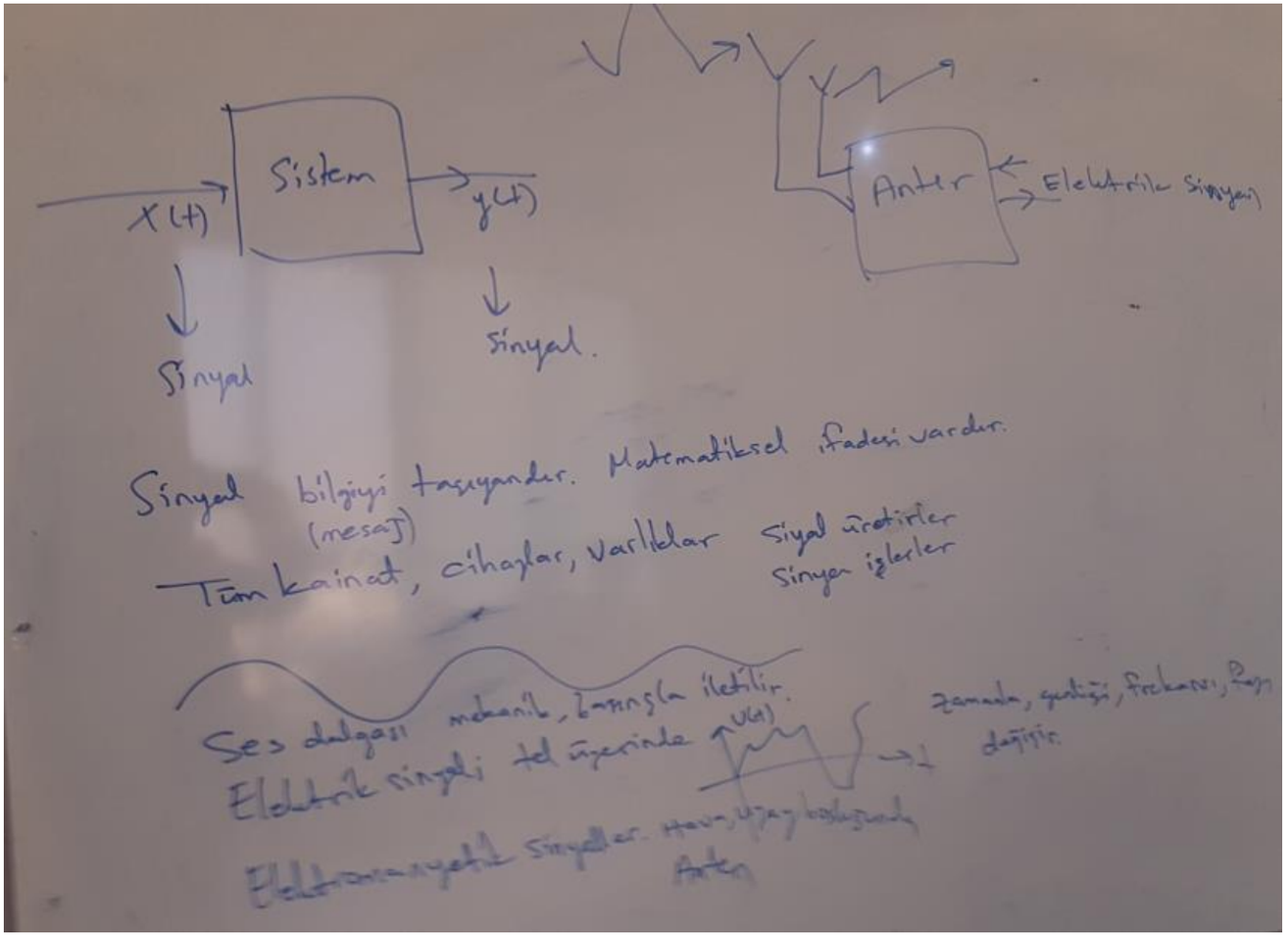
von Neumann Machine

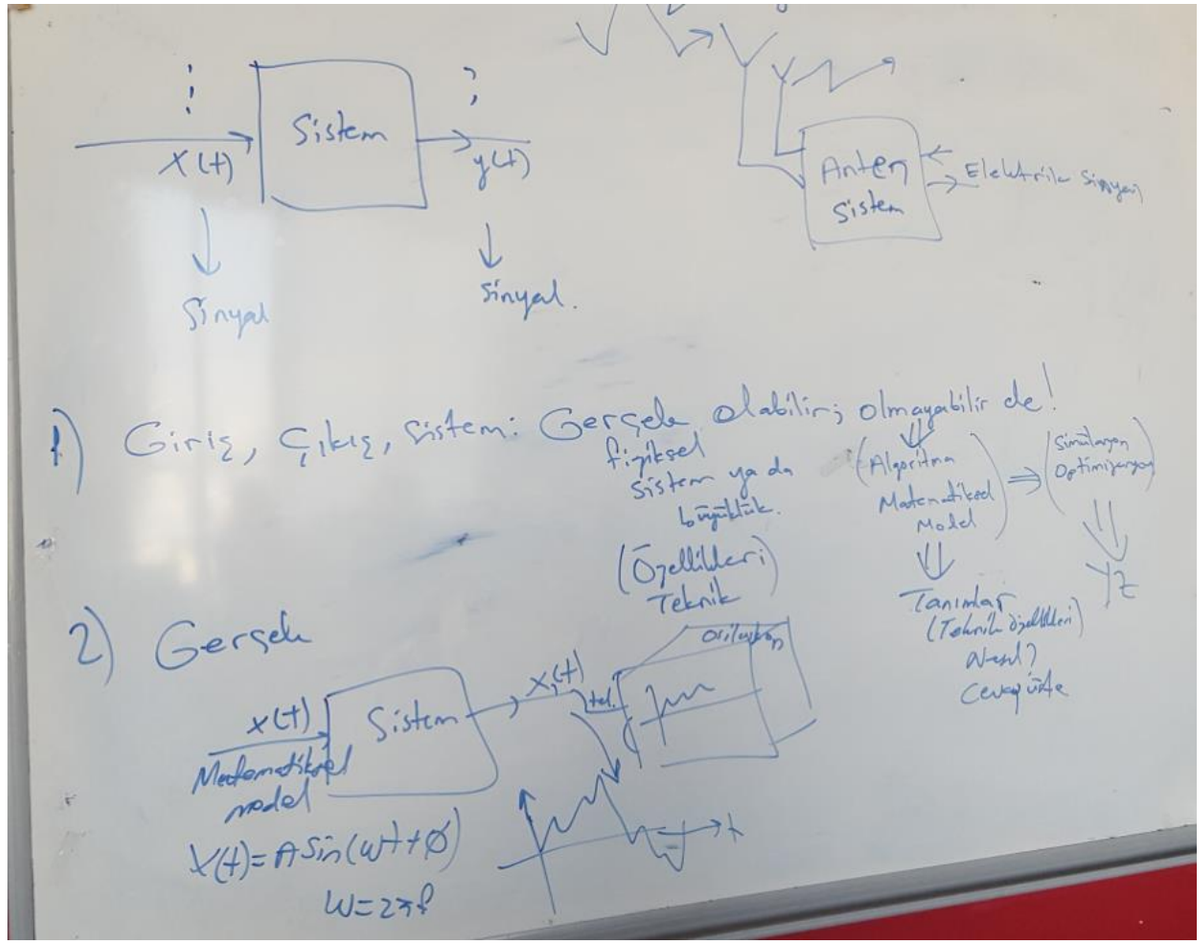
- Store programs in electronic memory along side the data (1943)
  - Move and manipulate a program like data
  - Enabled high-level programming languages



Günümüz Bilgisayar Sistemleri

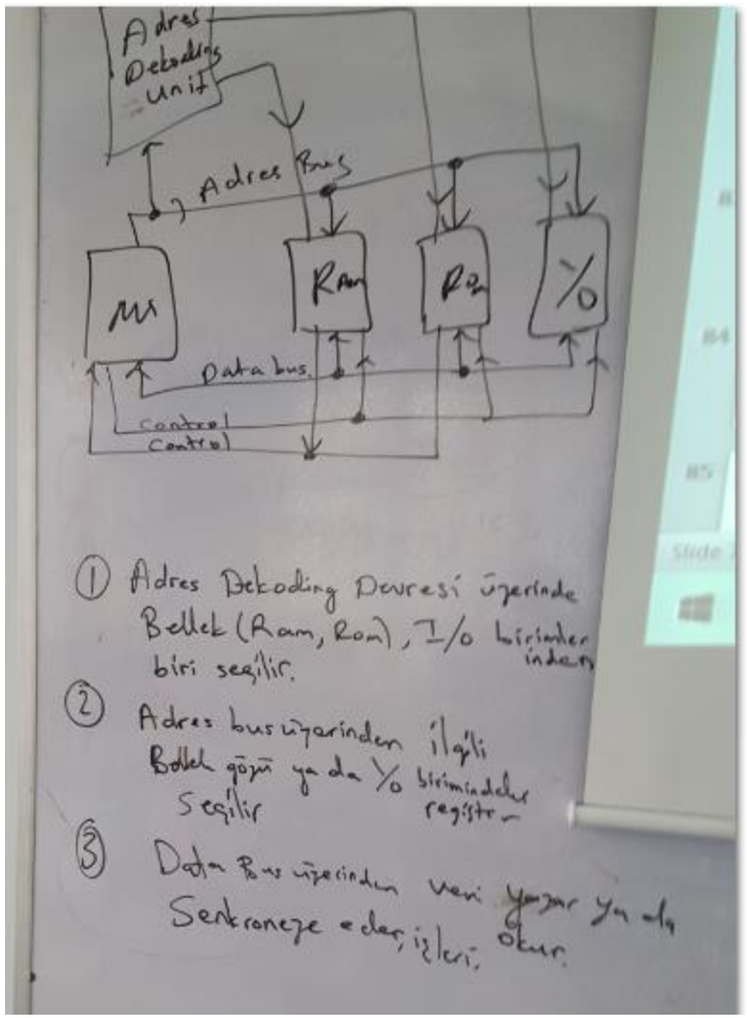
Sistemler ve Sinyaller





Bilgisayar

- Bilgisayar girişten verileri alan, işleyen, saklayan, isteğe uygun çıktı üreten (manipule) bir elektronik cihazdır.
- En temel elektronik devre elemanı transistördür. Atomitk yapıda üretilmektedir. Elektron akışını kontrol eder. İletkenden birim zamanda geçen elektrik yükü (elektron) miktarına Akım denir.
- Veri bit (0/1) olarak ikili sayı sisteminde saklanır, işlenir, iletilir. Bitler elektrik sinyaller ile temsil edilir.
- Dış dünyadaki tüm sinyaller analogtur. Bu nedenle analog sinyal dönüştürücüler kullanılır.
- Gelecekte quantum bilgisayarlar kullanılacaktır. Qubitler elektronlar ile temsil edilmektedir.



Bilgisayar Bileşenleri

- Mikroişlemci, Makine dili assemble
- Ana Bellekler: Ram (hem yazılan hem de okunan bellek, Veri) ve Rom (sadece okunan bellek, program kodları)
- I/O
- Clock and Timing
- Sistem Bus (Adres, Data, Kontrol): Mikroişlemci ile bellekler ve I/O birimleri arasında haberleşme yapan hatlardan oluşur.

Programlama

- Algoritma (Harazmi, Boolean Algebra) akış diyagramıdır, çözüm adımlarının fazlandırılmasıdır.
- Matematiksel model (Sabitler ve değişkenler)
- Programlama, insanlar ile mikroişlemci ve mikroişlemci ile bellekler ve I/O birimleri arasındaki etkileşimi sağlar.
- Yapay Zeka alanında gerekli programlama dilleri: Matlab (Bilimsel), Python, C++ (Donanım), Java Script (Web arayüzleri). Kütüphane denilen sınıfsal yazılımlar yardımıyla parogramların bütünleştirilmesi yapılır. Modüllerden sistemler oluşturulur.

Binary (ikili): 0/1 → Bit

Bilgisayar: Veri, işleyen, I/O

Transistor → Yarıiletken tabanlar.
elektron akışı kontrol edilir
— Anahtarlama
— Kuvvetlendirme.
— Şuanki durum ve bir sonraki duruma

Sembol (mesaj)

Veri (1/0)

Karakter (klavye tuşları) → ASCII → her karakter → 8 bit

Renk tonları → 8 bit / 16 bit / 32 bit / ...

$(278)_d = (b)_2 = 2^8 + 2^4 + 2^2 + 2^1$

En yüksek $2^{\wedge}$

maksimum indis

olanlar → 1, olmayanlar → 0

8 7 6 5 4 3 2 1 0
$(1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_b$

$(1\ 0001\ 0110)_b = (d)_?$

8 7654 3210

$= 2^8 + 2^4 + 2^2 + 1$

Sağdan soldan indisle olanlara 2^n Topla.

256
16
4
2
$\Rightarrow (278)_d$

278
256
022
16
6
4
2

Makine Öğrenmesi Algoritma

- Matematiksel Modelleme
- Simülasyon
- Optimizasyon
- Kalibrasyon
- Akıllı Algoritmalar

Matematiksel Modelleme

- Uygulamalı matematik
- Lineer Cebir: Vektör, Matris, Özdeğer, Özvektör
- Boolen Algebra
- Türev, İntegral, Limit
- Diferansiyel denklemlerde faz düzlemleri
- Olasılık, Bayeas, Markov zincir, Kolmogorov; Hipotez, Olasılık dağılım fonksiyonları
- Tamamlayıcı istatistik: Standart sapma, varyans, ağırlık ortalama

Mikroişlemci

- Mikroişlemci temel bileşenleri: Registers (Geçici kayıt ediciler), ALU (Aritmetik Lojik birimi), Kontrol birimi, Sistem Bus (Adres, data ve kontrol)
- Metinler, Resimler, Video, Sesler, Rakamlar, Harflerin tümü bit olarak ikili sayı sisteminde temsil edilir.
- ASCII kodu ile klavyedeki tüm tuşlar 8 bit olarak temsil edilir. Toplam=2^8=256 adet
- Bellekler byte (8bit) boyutu ile temsil edilir. 2^40byte= 1 Terabyte
- Veri işleme ve veri transferi 10^n bit/saniye ile temsil edilir.

ALU

- Tüm işlemler bit olarak ikili sayı sisteminde yapılır.
- Aritmetik işlemlerden toplama ve sağa ya da sola öteleme yapılır. Çıkarma toplama işleminin tersidri. Sağa ve sola öteleme ile çarpma ve bölme işlemleri yapılır.
- Mantıksal işlemler: OR, AND, NOT, NOR, NAND, XOR, XNOR
- Karşılaştırma: >,<, ≥, ≤, =, ≠, ...
- Yakın gelecekte ALU olarak quantum hesaplama çalışacaktır.

Veri Tipleri

VERİ TİPLERİ

Int Tip

- Integer = Tamsayı
- Tamsayıları içerir. Bellekte 2 Byte tutar.
- 5 , -19 , 25000 gibi
- Minimum : -2^{31} = -32768
- Maksimum : $2^{31}-1$ = 32767

Long tip

Uzun tamsayı. Bellekte 4Byte tutar.

Minimum :-**2,147,483,648 .**

Maksimum :**2,147,483,647**

VERİ TİPLERİ

Gerçel Tipler (Float, Double)

- Gerçel sayıları içerirler.

float : Bellekte 4 Byte yer tutar.

- 3.4E-38 ile 3.4E+38 aralığında değer alır.
- Hassasiyet 7-8 basamaktır.

double : Bellekte 8 Byte ter tutar.

- 1.7E-308 ile 1.7E308 aralığında değer alır.
- Hassasiyet 15-16 basamaktır.

- NOT: Bilimsel gösterim biçimi $2.5*10^{3}$ = 2.5E3 $2.5*10^{-3}$ = 2.5E-3

54

VERİ TİPLERİ

- **Char Tip**
- Char : Karakter :
- Alfanumerik karakterleri içerir.
- '5' , '*' , 'K‘ gibi

55

Kod Sistemleri

Bilgisayarlar yalnızca sayılarla çalışırlar, oysa bizim harflere ve diğer sembollere de gereksinimimiz vardır. Bu semboller de sayılara karşılık düşürülecek biçimde kodlanırlar. Program örneğin bu sayı ile karşılaşırsa ekrana karşılık düşen sembolü basar, yada klavyeden gelen sayının sembolik karşılığını , yazıcıdan çıkarır.

Bir çok kodlama türü olmasına karşın dünyada bilgisayar ortamlarında ANSI tarafından 1963 yılında standartlaştırılan **ASCII**(**A**merican **N**ational**C**ode for **I**nformation **I**nterchange) kodlaması yoğun olarak kullanılmaktadır. Ancak günümüzde , ASCII kodları çok dilliği sağlayabilmek için yetersiz kaldığından UNICODE kodlaması yaygınlaşmaktadır. Ancak pek çok uygulamada ASCII kodlaması hala geçerliliğini korumaktadır.

ASCII temel olarak 7 bit' tir. 127 karakterden oluşur. Ama Extended kısmıyla birlikte 8 bit kullanılmaktadır. Ancak genişletilmiş kısımdaki semboller yazılım ortamına göre değişebilmektedir.

ASCII ilk 128 Sembol

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ASCII genişletilmiş kısım

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Yazılım

YAZILIM GELİŞTİRME

Yazılım : değişik ve çeşitli görevler yapma amaçlı tasarlanmış elektronik aygıtların birbirleriyle haberleşebilmesini ve uyumunu sağlayarak görevlerini ya da kullanılabilirliklerini geliştirmeye yarayan makine komutlarıdır.

Yazılım, elektronik aygıtların belirli bir işi yapmasını sağlayan programların tümüne verilen isimdir. Bir başka deyişle, var olan bir problemi çözmek amacıyla bilgisayar dili kullanılarak oluşturulmuş anlamlı anlatımlar bütünüdür.

YAZILIM GELİŞTİRME SÜRECİ

Bilgisayar yazılımları genel olarak 3 ana grupta incelenebilir. Bunlar;

1- Sistem Yazılımları (System Software): Bilgisayarın kendisinin işletilmesini sağlayan, işletim sistemi, derleyiciler (compilers) (Yazılım programında, yazılan programı makine diline çeviren program), çeşitli donatılar (facility) gibi yazılımlardır. Windows, Pardus, Android, vb.

2- Uygulama Yazılımları (Application Software): Bu kullanıcıların işlerine çözüm sağlayan örneğin çek, senet, stok kontrol, bordro, kütüphane kayıtlarını tutan programlar, bankalardaki müşterilerin para hesaplarını tutan programlar vs. gibi yazılımlardır. Örneğin Winrar, Word, Messenger,vb.

3- Çevirici Yazılımlar: Herhangi bir dilde yazılan programı makine diline çeviren yazılımlardır. Örneğin C , Pascal, Java,vb.

YAZILIM GELİŞTİRME SÜRECİ

Bir yazılımı geliştirmek için temel olarak şu adımları uygulamak gerekir;

1. Analiz: Gereksinimlerin belirlendiği, çözümlendiği ve çerçevelendiği aşamadır. Bu aşamada yazılımın ne yapacağı, hangi ihtiyacı karşılayacağı hangi problemi çözeceği belirlenir.
2. Tasarım: Analizle belirlenen yazılımın en uygun şekilde nasıl gerçekleştirilebileceğinin belirlenmesidir. Müşterinin gereksinimlerine ve koşullara bakılarak hangi programlama dili, teknoloji, mimari, araç vb. kullanılarak, çözümün planının ve mimarisinin tasarlanmasıdır.
3. Geliştirme: Tasarımın artık hayata geçirilmeye başlandığı aşamadır. Kodlama bu aşamada yapılır. Bu aşamada arayüz tasarımları ve çeşitli ayarlamalar da yapılır.
4. Hatalardan Arındırma: Geliştirme aşamasında ortaya çıkan arayüz, kod, veritabanı, doküman gibi ürünlerin istenilen şeye uygun olup olmadığı test edilir. Eğer yazılımın çeşitli bölümlerinde hatalar bulunuyorsa,bu hatalar düzeltilerek yeniden test edilir.

YAZILIM GELİŞTİRME SÜRECİ

Algoritmalar yazılım geliştirme sürecinde, programlamayla tasarım arasında bir yerde kalırlar. Büyük projelerde bazen yazılım mimarları karmaşık bir işin çözümü için önceden çalışarak çözümü bulur ve bunun algoritmasını oluştururlar.

Daha sonrada yazılımcı bu algoritmayı alarak programlar. Orta ve küçük çaplı projelerde yazılımcı kendi algoritmasını kendisi belirler ve programını buna göre yazar.

Günümüzde artık standart iş yazılımları geliştirilirken yapılan işler için algoritma yazmadan direkt programlama yoluna gidilmektedir. Fakat karmaşık problemleri çözümlemek için mutlaka algoritmalara başvurulması gerekir.

YAZILIM GELİŞTİRME SÜRECİ

Yazılım geliştirme sürecinde bir programcının ihtiyacı olan bilgi alanları şunlardır;

Programlama Dili: Eğer ki bir yazılım geliştirilecekse, en azından bir programlama diline hakim olunması gerekir.

Yazılım Geliştirme Arabirimi: Arayüz tasarlayıcı, kod düzenleyici, derleyici ve yorumlayıcıyı bir arada barındıran ve yazılım geliştirme sürecini kolaylaştıran araçlara verilen isimdir. İyi bir programcının mutlaka bir yazılım geliştirme arabirimine hakim olması gerekir.

Platform: Geliştirilen yazılım, hangi ortamda çalıştırılacak ise (Windows, Web Browsers, MsDos,vb) bu platformların kendine has kullanılan komutlarının iyi bilinmesi gerekir. Örneğin internet ortamında çalışacak bir yazılım geliştirilecek ise oturum yönetiminin iyi bilinmesi gerekir.

Teknoloji: Üzerinde çalışılan yazılım hangi teknolojileri kullanılıyor ise programcının bu araçlara da hakim olması gerekir. Örneğin bir email gönderme – alma işlemi yapacak bir yazılım geliştirilecekse, SMTP protokolünün iyi bilinmesi gerekir.

YAZILIM GELİŞTİRME SÜRECİ

En İyi Pratikler: Belirli bir teknolojide bir çözüm üretmek isteniyorsa, öncelikle bu işi birileri yapmış mı diye bakmak önemlidir. Bu işlem başka bir yazılımdan kopya çekmek anlamında değil, sistemin nasıl çalıştığı hakkında bilgi edinip bir fikir edinme hakkında ciddi avantajlar sağlar.

Algoritma: Geliştirilen yazılım içerisindeki karmaşık bir problemin çözümünün bulunmasını sağlar. Yapılacak işin en iyi nasıl çözüleceğini bulmak için algoritmalardan faydalanılır.

PROGRAMLAMA DİLLERİ

Programlama dili: Yazılımın bir algoritmayı ifade etmek amacıyla, bir bilgisayara ne yapmasının istendiğinin anlatıldığı bölümdür.

Günümüzde en yaygın olarak kullanılan programlama dilleri;

- C, C++, C#.NET, Objective C
- Pascal, Delphi
- QBasic, Visual Basic.NET
- Java, Netbeans
- Html
- Php, Asp, Asp.NET
- Sql,PL/SQL

PROGRAMLAMA DİLLERİ

Programlama dilleri ile ilgili ortak kavramlar;

Kaynak Kod (Source Code): Bir programın oluşması için program dili ile yazılan metinlere kaynak kod adı verilir.

Kod Düzenleyici: Herhangi bir programlama dilinde program yazmak için, kodla ilgili ipuçları veren, hataları tespit eden ve hataların düzeltilmesi için çeşitli uyarı mesajları veren uygulamaya kod düzenleyici denir.

Derleyici (Compiler): Herhangi bir programlama diliyle yazılmış olan kaynak kodunu makine dönüştüren programlara derleyici denir.

Yorumlayıcı (Interpreter): Kaynak kodunun satır satır, komut komut derleyerek makine diline çeviren ve çalıştıran programlara yorumlayıcı adı verilir. Yorumlayıcının amacı, programcının yazdığı programı satır satır işleterek, çalışmasını izlemesini ve varsa hatalarını bularak düzeltilmesini sağlamaktır.

Some Common Complexity Classes

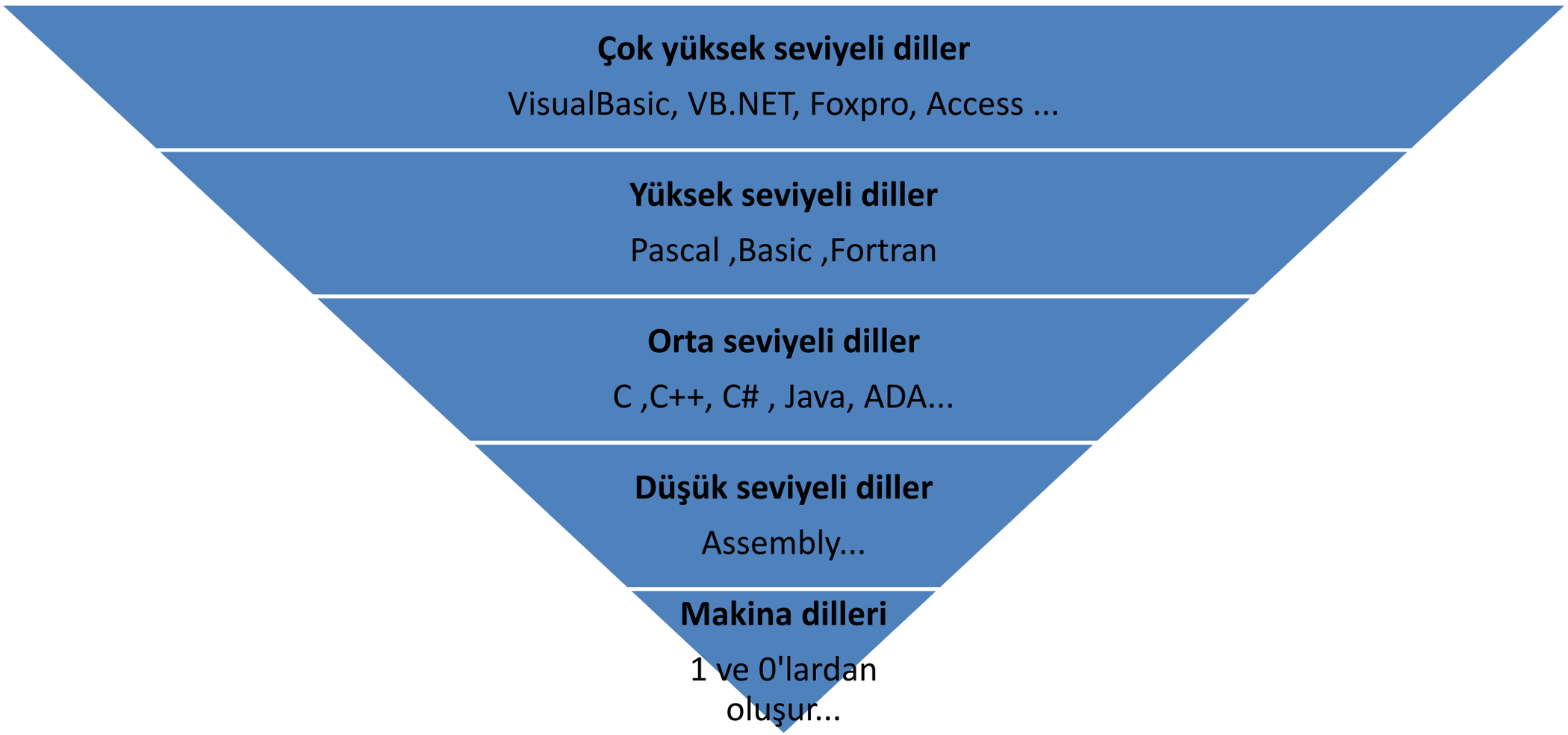
O-notation	Adjective Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n \log n)$	n log n
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential
$O(10^n)$	Exponential
$O(2^{2^n})$	Doubly exponential

Time Complexity

- We have to consider the following cases:
  - Worst case
  - Best case
  - Average case

- Proof
- Analysis
- Complexity
- Recursive Solution

PROGRAMLAMA DİLLERİNİN SINIFLANDIRILMASI



Alt Düzeyli Diller

İkilik sayı (Binary) sisteminde (0,1) kodlanabilen dillerdir. Örneğin , makine dili ve assembly.

Makine Dili : Sadece ikili sayı sisteminde kodlanan ve bilgisayarın doğrudan yorumlayıp, işleyebileceği tek programlama dilidir**.**

Makine dili bilgisayarın ana dilidir.

Makine dili dışındaki tüm diller semboliktir ve makine diline dönüştürülmesi gerekir.

Assembly : Makine diline en çok benzeyen dildir. **0** ve **1** yerine **MNOMENIC** denilen semboller kullanılır. Bunlar **ADD, MOV, JMP, STR** gibi sembolik komutlardır.

Örneğin ADD A,B ; A ve B adresindeki bilgileri toplayıp sonucu B adresine yerleştirir.

Assembly dilinde yazılan her program çevirici denilen ASSEMBLER'den geçirilerek makine diline çevrilir.

- Dilden dile çevrim

Programlama dili ile yazılan program (kaynak kodu)(**source code**)

Çevirici program

- **Assembler**
- **Compiler**
- **Interpreter**

Makine diline Çevrilmiş kod (**object code**)

CPU tarafından işlenir

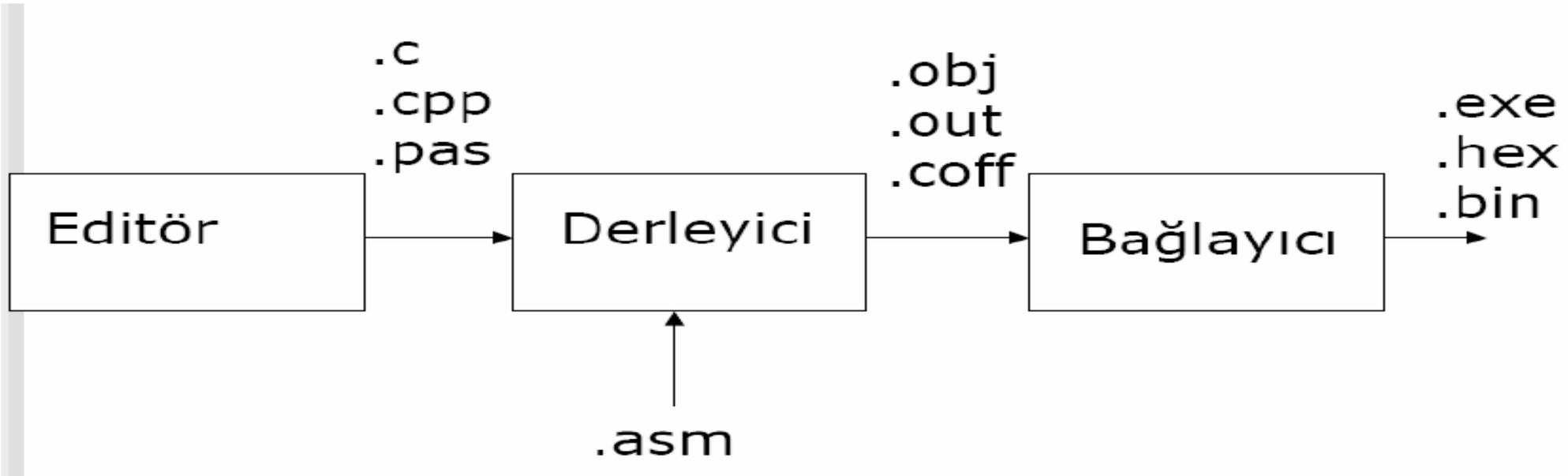
Programlama Dili Nedir?

- Programlama Dili, istenilen hesaplamaları yapmak için, elde edilen veriyi saklamak için ve girdi/çıktı aygıtlarına veri gönderme/alma gibi işlemleri yapmak için kullanılan dildir.
- Doğal dillerde olduğu gibi programlama dillerinde de belirli bir yazım kuralı (sentaks) vardır.
- Programlama dilleri ile sadece bilgisayarlar üzerinde çalışan uygulamalar değil, işlemcisi ve belleği bulunan diğer elektronik cihazlarda çalışan uygulamalar da yazılır.

Derleyici Nedir?

- **Derleyici (Compiler**), bir bilgisayar dilinde yazılmış olan kodu, bilgisayarın (yada elektronik cihazın) donanımına uygun makine diline çeviren bilgisayar programıdır.
- Derleyici öncelikle yazılan program kodunun doğru yazılıp yazılmadığını kontrol eder, eğer hatalar varsa bunları programcıya bildirir.
- Eğer kod doğru ise derleme yapılan sisteme uygun olan 0 ve 1'lerden oluşan makine kodunu üretir (EXE dosyası).

Yazılım Geliştirme Araçları

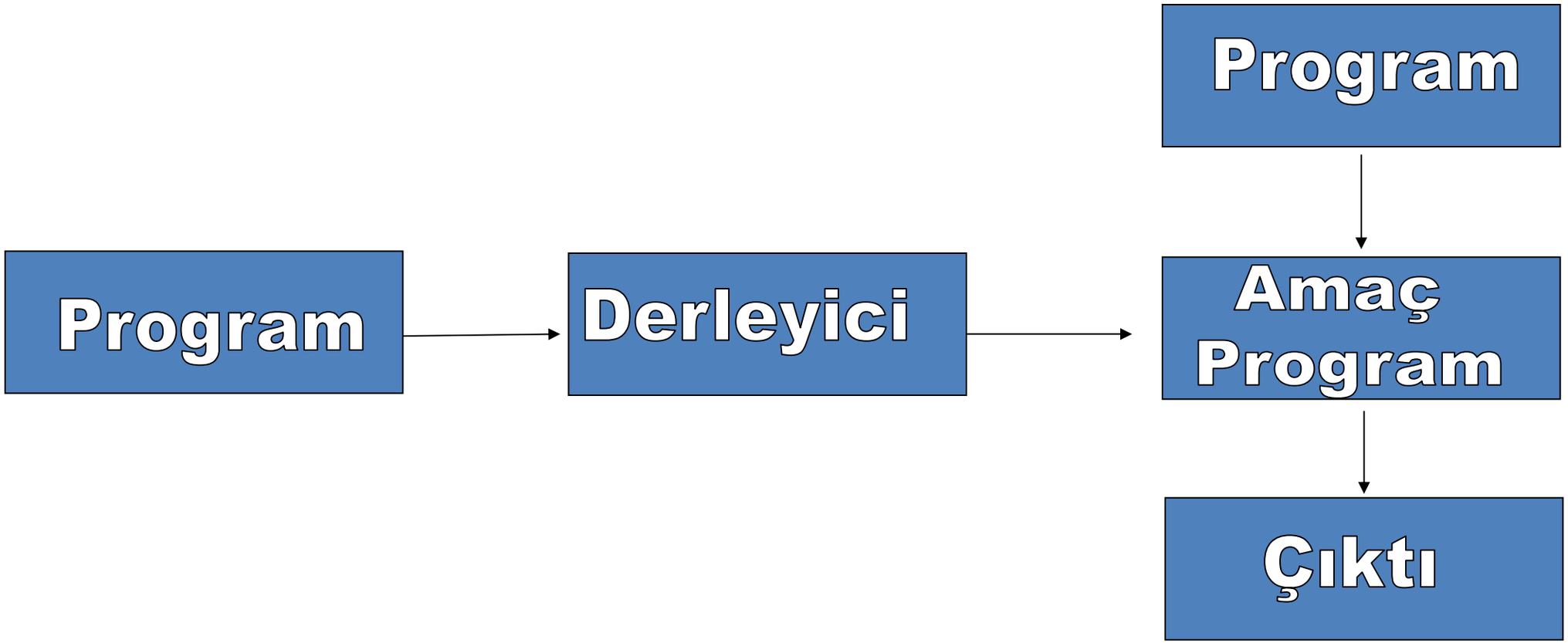


Derleyiciler

- Derleyicisi(Compiler) olan dillerde yazılan program (kaynak program) derleyiciden geçirilerek makine diline dönüştürülür.
- Bu derleme sırasında yazım hatası, sayısal hata, komut sıra hatası, döngü hatası vb. Gibi hatalar varsa bu hatalar listelenir.
- Programcı bu hataları gidererek yeniden derler.Bu tür programlar ancak bütün olarak derlendikten sonra çalıştırılabilir.

Derleyici, programın makine kodunu bir kez oluşturarak ayrı bir dosyaya kaydeder.Program her çalıştırılışta bu kod otomatik olarak kullanılır.

C, PASCAL, COBOL derleyicisi olan üst düzey dillerdir.



Derleyici ile programlama

Yorumlayıcılar

- **Yorumlayıcılar(Interpreter) da** yazılan programları makine diline dönüştüren yazılımlardır.
- Ancak bu dönüşüm derleyicilerden farklı olarak gerçekleştirilmektedir.
- Yorumlayıcılar her satırı anında makine diline çevirerek çalışır ve bu kodu dosyaya kaydetmez.
- Dolayısıyla program her çalıştırıldığında her satır yeniden makine koduna dönüştürülür.Bu yüzden yorumlayıcılar yavaş çalışmaktadır.BASIC ve DBASE hem derleyicisi hem de yorumlayıcısı olan üst düzey dillerdir.

Yorumlayıcı ile Programlama

Üst Düzeyli Diller

- Programcının makineye olan bağımlılığını ortadan kaldırmak için geliştirilmişlerdir. Alt düzeyli dillere göre öğrenmesi ve program yazımı kolaydır.
- Komutlar, genellikle İngilizce kelimelerden veya bu kelimelerin kısaltılmasından oluşturulur.
- Bilgisayar sadece ana dili olan makine dilinden anladığından dolayı, üst düzey dilllerle yapılan programların çalışabilmesi için makine diline dönüştürülmesi gerekiyor.İşte bu dönüştürmeyi **Derleyici** (compiler) ve/veya **Yorumlayıcı (Interpreter)** yapmaktadır.

Coding and Languages

Kodalama

- Tim Cook: “İkinci bir dil öğrenirken muhakkak kodalama da öğrenmelisizi... Gelecekte kodlamayı öğrenmek, İngilizce öğrenmekten daha önemli olacaktır. Kodlama, insana kendinizi ifade edebileceğiniz bir dil.»

- Büyük bir iş arama web sitesi olan Glassdoor, son zamanlarda Cook’un önerilerini destekleyen bir rapor yayınlamıştı. Raporda, web sitesinde en çok ödeme yapılan işlerin üçte birinin programlama öğeleri gerektirdiği belirtilmişti.

- Ancak Cook, kodlama öğrenmenin yararlarının yüksek ücretlerin çok daha ötesinde olduğunu, sadece bilgisayar bilimcilerin değil herkesin kodlama bilmesi gerektiğini söyledi. Kodlamanın temelinde yaratıcılık ve tasarım kabiliyeti yattığı için kodlama bilmek çalışanların yaptığı işlerde yenilikçi yaklaşımlarda bulunabilmelerini sağlıyor.

Programlama Dillerinin Seviyelerine Göre Sınıflandırılması

- Seviye, bir programlama dilinin insan algılamasına olan yakınlığının bir ölçüsüdür**.**
- **Yüksek seviyeli diller** insan algılayışına daha yakın, alçak seviyeli diller de bilgisayarın doğal çalışmasına daha yakın olan dillerdir.
- Dillerdeki seviye yükseldikçe programcının işi de kolaylaşır. Öyle ki, çok yüksek seviyeli programlama dillerinde artık bir işin nasıl yapılacağına ilişkin değil, ne yapılacağına ilişkin komutlar bulunur.
- Seviyenin yükselmesi programcıya kolaylık sağlamakla birlikte genel olarak verimliliği ve esnekliği de azaltır.

86

Programlama Dillerinin Tarihçesi

- 1962 - APL
- 1964 - BASIC
- 1964 - PL/I
- 1970 - Pascal

→ Yapısal programlama

- 1970 - Forth
- 1972 - C
- 1972 - Prolog
- 1978 - SQL
- 1983 - Ada
- 1983 - C++
- 1987 - Perl

→ Nesne yönelimli dillerin ortaya çıkışı

1990 lar, Internet

- 1991 - Python
- 1991 - Java
- 1995 - PHP
- 2000 - C#

Tamamı nesne yönelimli dillerdir. Yeni programlama kavramlarından ziyade, programlamanın kolaylaşmasını ve taşınabilirliği amaçlamaktadırlar

- Çok Yüksek Seviyeli Programlama Dilleri ya da Görsel Diller (FOXPRO, PARADOX, ACCESS.., VISUAL BASIC, IV.KUŞAK DİLLER)
- Yüksek Seviyeli Programlama Dilleri (PASCAL, COBOL, FORTRAN, BASIC,...)
- Orta Seviyeli Programlama Dilleri (C)
- Alçak Seviyeli Programlama Dilleri (Sembolik Makine Dilleri)

PROGRAM - ALGORİTMA –AKIŞ ŞEMASI

- **Program** : Belirli bir problemi çözmek için bir bilgisayar dili kullanılarak yazılmış deyimler dizisi.
- **Algoritma** bir sorunun çözümü için izlenecek yolun tanımıdır
- **Akış şeması** belirli bir işin yapılabilmesi için, basit işlemlerle şema halinde gösterilmesidir. Kısaca algoritmanın şemalarla gösterilmesidir.
- Algoritma geliştirildikten sonra, daha iyi anlaşılabilir olması ve programlama dillerine aktarımı daha kolay olması nedeniyle, akış şeması haline getirilir.
- Böylece sorunun çözüm basamakları, birbirleri ile ilişkileri ve bilgi akışı daha kolay görülebilir ve yanlışlıklar düzeltilebilir.
- Algoritma, bir problemin çözümünde izlenecek yol anlamına gelir.
- Algoritma, bir problemi çözmek için art arda uygulanacak adımları ve koşulları kesin olarak ortaya koyar. Herhangi bir giriş verisine karşılık, çıkış verisi elde edilmesi gereklidir.
- Matematiksel modelin çözülmesinde algoritmalar çok sık kullanılır.

89

PROGRAMIN ÇALIŞMASI

source ---→ compiler -----> object -----> link

kaynak derleyeci amaç bağlama

- kaynak kod : C dili ile yazılmış olan program.
- derleyeci : Kaynak kodu makina koduna çevirir
- amaç kodu : Kaynak kodun makina dilindeki karşılığı
- bağlama : Birden fazla amaç kodu dosyasının tek dosyada birleştirilmesi

90

- Her dilin mutlaka bir derleyicisi veya yorumlayıcısı vardır. Bazı dillerin ise hem derleyicisi hem de yorumlayıcısı vardır.

Fortran

İngilizce FORmula TRANslation kelimelerinin ilk hecelerinden türetilen FORTRAN, bilimsel hesaplamalar yapmak için geliştirilmiştir. Birkaç Sürümü vardır (FORTRAN IV, FORTRAN 77 VE FORTRAN 90)

Cobol

İngilizce COmmon Business Orianted Language kelimelerinin kısaltılarak adlandırılmasıyla oluşturulan COBOL ticaret işlemleri için geliştirilmiştir. En büyük özelliği komutlarının İngilizce'ye yakın olmasıdır.

Basic

Beginner's All Purpose Symbolic Instruction Code kelimelerinin baş harflerinden oluşturulmuş BASIC, eğitim amaçlı bir program olarak geliştirilmiştir. Fakat ticari ve bilimsel sahalarda da kullanılır ve oyun programları yazılabilmektedir.

Quick Basic, Turbo Basic, Gw basic, Visual Basic, gibi derleyicileri vardır.

Pascal

Fransız Matematikçisi Blaise Pascal'ın adını taşıyan PASCAL İsviçre'li Niklaus Wirth tarafından programcılığı öğretmek amacı ile geliştirilmiştir.Günümüzde iş ve bilim çevrelerinde yaygın olarak kullanılmaktadır. En çok kullanılan sürümü ise Borland firmasının Turbo Pascal'ıdır.

C

Dennis Ritchis tarafından Bell laboratuvarlarında geliştirilmiştir.C'nin makine diline çevrilmesi diğer üst dizey dillere göre daha kolaydır. En çok kullanılan dillerdendir. Bir ağ işletim sistemi olan UNIX C ile yazılmıştır.

C NASIL BİR DİL?..

- **Orta seviyeli bir dildir.** Yazılan *C* kodu ile makina kodu arasında bağlantı kolaylıkla kurulabilir.
- **Sistem programlama dilidir.** Bugün işletim sistemleri, derleyiciler, editörler gibi sistem programlarının hemen hepsi yoğun olarak C kodu içermektedir. Ancak sistem programlamanın dışında da birçok uygulamada C verimli olarak kullanılabilir.
- **Algoritmik bir dildir.** Yalnızca dilin sintaks ve semantik yapısını bilmek yetmez. Problemleri çözebilecek bir algoritma bilgisine de ihtiyaç duyulur.
- **Diğer diller arasında taşınabilirliği en fazla olanlardan biridir.**
- **İfade gücü yüksek ve okunabilirlik özelliği kuvvetli bir dildir.**
- **Çok esnektir.** Bu yüzden programcının hata yapmayacak bir bilgiye ve deneyime sahip olması gerekir.
- **Atomik bir dildir.** Cde altprogramlama tekniği ileri düzeyde kullanılmaktadır.
- **Güçlü bir dildir.** Tasarım özellikleri çok iyidir. Cye ilişkin yapıların ve operatörlerin bir kısmı daha sonra diğer diller tarafından da benimsenmiştir.
- **Verimli bir dildir.** C programları seviyesi dolayısıyla daha hızlı çalışır.
- **Doğal bir dildir.** C bilgisayar sisteminin çalışma biçimiyle uyum içindedir.
- C++ ile nesne yönelimlilik özelliğine de sahip olmuştur.

97

İLK PROGRAM

```
#include<stdio.h>
#include<conio.h>
main()
{
        printf("Bu bir satirlik yazidir.");
}
```


98

Programming Language

- First-generation: Machine language
- Second-generation: Assembly language
- Third-generation: High-level language
- Fourth-generation (Fifth-generation)

1GL: Machine language

- A set of primitive instructions built into every computer
- The instructions are in the form of binary code
  - 1101101010011010

2GL: Assembly language

- Low-level programming language to represent machine-language instructions
  - E.g.: MOV AL, 8h
- Assembly code need to be converted into machine code by using an assembler
- Assembly program
  - is platform dependent
  - Combination of mnemonic and machine instruction

3GL: High-level language

- English-like and easy to learn and program.
- E.g.:
  - Area = 5 * 5 * 3.1415;
- COBOL, FORTRAN, BASIC, Pascal, Ada, C, Visual Basic, Delphi, C++, C#, Java
- Source program is compiled into machine code by a compiler and linked to supporting library code by a linker to form an executable file.

4GL / 5GL

- 3GL offered greater power to the programmer, while 4GL open up the development environment to a wider population. (Applications Development Without Programmers)
- Database query languages: SQL…
- Data manipulation, analysis, and reporting languages: MATLAB, SPSS…

Category (3GL)

- Windows Application
  - C, C++, Java, Visual Basic, C#
- Web Application
  - Server Side
    - PHP, JSP (Java), ASP.NET (Visual Basic, C#), …
  - Client Side
    - JaveScript, VBScript

The Binary Machine

- A modern computer can run programs written in JavaScript, Pascal, Visual Basic, Visual C++, etc.
- However, computers can only understand one language: the machine language it is not easy to use.
- The machine language of a Sun workstation is different from a PC (or other platform), however, they can run the same C++ program.

History of C language

- C programming is a general-purpose, procedural, imperative computer programming language.
- It was developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system.
- C is the most widely used computer language.

Why to Learn C Programming?

- **C programming** language is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Software Development Domain. Following are the key advantages of learning C Programming:
- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

Structure of a C Program

Hello World Example

A C program basically consists of the following parts −

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

```
#include <stdio.h>
int main()
{
        /* my first program in C */
        printf("Hello, World! \n");
        return 0;
}
```

C and C++

- C is arguably the most successful programming language of all time
  - Ama nasıl karar vereceksin?
    - Yazılan program sayısı
    - Yazılan programların önemi
    - Programcı sayısı
    - Ömür
    - Diğer diller üzerindeki etkisi
    - Faydalar/geliştirme maliyetiNumber of programmers
  - Alternatives
    - Fortran
    - Cobol
    - Lisp
    - C++
    - Java
    - PHP
    - Python
    - …

Machine Languages

- Only language computers directly understand
- “Natural language” of computer
- Defined by hardware design
  - Machine-dependent
- Generally consist of strings of numbers
  - Ultimately 0s and 1s
- Instruct computers to perform elementary operations
  - One at a time
- Cumbersome for humans
- Example:

```
+1300042774
+1400593419
+1200274027
```

Assembly Languages

- English-like abbreviations representing elementary computer operations
- Clearer to humans
- Incomprehensible to computers
  - Translator programs (assemblers)
    - Convert to machine language
- Example:

```
LOAD        BASEPAY
ADD         OVERPAY
STORE       GROSSPAY
```

High-level Languages

- Similar to everyday English, use common mathematical notations
- Single statements accomplish substantial tasks
  - Assembly language requires many instructions to accomplish simple tasks
- Translator programs (compilers)
  - Convert to assembly language
- Interpreter programs
  - Directly execute high-level language programs
- Example:

```
grossPay = basePay + overTimePay
```

Programming Approaches

- Structured programming (1960s)
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - Focus on what the program does
- Object Oriented programming
  - Object is an entity characterized by a *state* and a *behavior*
    - state is encoded in the computer program as *data*
    - behavior is encoded as methods

Objects

- Reusable software components that model real world items
- Meaningful software units
  - Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.
  - Any noun can be represented as an object
- More understandable, better organized and easier to maintain than structured programming
- Favor modularity
  - Software reuse
    - Libraries

C++

- C++ programs
  - Built from pieces called classes and functions
- C++ standard library
  - Rich collections of existing classes and functions
- “Building block approach” to creating programs
  - “Software reuse”

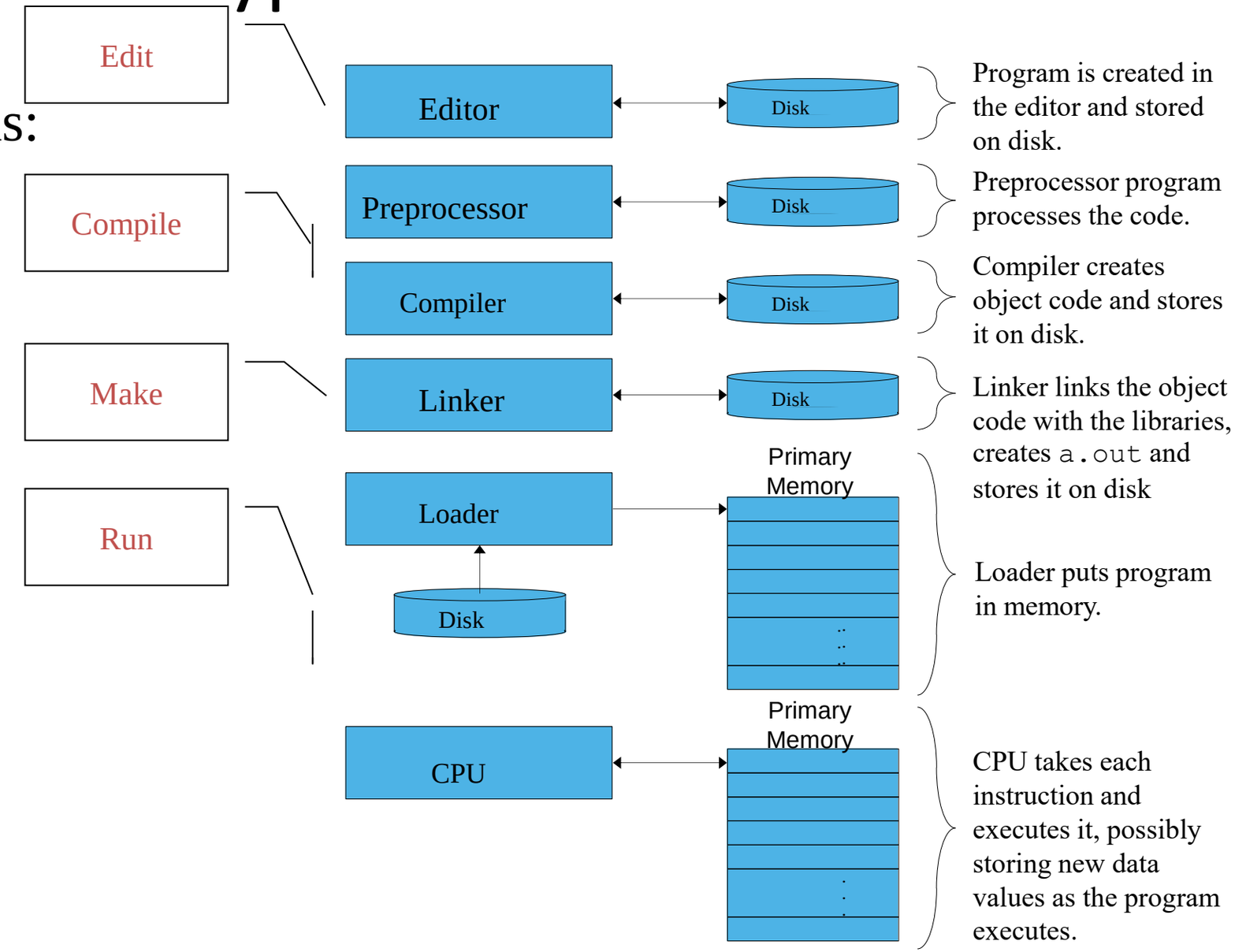
Basics of a Typical C++ Environment

- C++ systems
  - Program-development environment
  - Language
  - C++ Standard Library

Basics of a Typical C++ Environment

Phases of C++ Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Computational Methods of Scientific Programming

- Languages to be covered:
  - Fortran, Matlab, Mathematica, C, C++, Python and graphics and advanced (parallel and GPU computing) topics

Language concepts Fortran, C, C++

- Compiled languages i.e., source code is created with an editor as a plain text file.
- Programs then needs to be “compiled” (converted from source code to machine instructions with relative addresses and undefined external routines still needed).
- External routines are those needed to do such operations as read disk files, write to the screen, read the keyboard strokes etc.)
- The compiled routines (called object modules) need then to be linked or loaded.

Fortran, C, C++ cont.

- Linking creates an executable with relative addresses resolved and external routine loaded from the system and user libraries.
- The executable can then, in most cases, be run on any machine with the same architecture.
- Compiling and linking can be done in one user step.

Fortran, C, C++ cont.

- *Advantages:* Programs are fast and can be run on many machines that don't need to have Fortran or C++ loaded.
- (There are exceptions to this depending on linking options. Safest is "static linking" in which case system libraries are loaded into program. Dynamic linking expects to find specific versions of system routines in specific locations.)

MatLab

- Interactive language with some automatic compiling (into MatLab pseudocode) of user subroutines (M-files).
- User programs can be developed as scripts
- Speed of modern processors makes this type of approach reasonable.
- Graphics and Graphical user interfaces (GUI) are built into program (for Fortran, C, and C++ graphics must be done through external or user written routines).

Matlab continued

- *Advantages:* Since interactive, user can debug on the fly, results available immediately (i.e., values of specific variables can be viewed at any time. In Fortran, C++ code must be changed to output values or a debugger program used). Integrated environment which can guide program development and syntax
- *Disadvantages*: Code can only be exported to users with the same version of Matlab but can often be used on different platforms depending on whether platform specific options are used). Code can be often converted to C and compiled (license needed)

Mathematica

- Interactive symbolic manipulation program with built in computation and graphics.
- This type of program is often used to derive algorithms for MatLab, Fortran and C++ but can also be used to generate results.
- Work can be organized into “work-books” that can be extended as a project progresses
- Program has options to export code and formulas in word processing languages

Mathematica

- *Advantages:* Symbolic manipulation so that it yields formulas rather than numerical results. Analysis of equations gives insights into characteristics of problems.
- Can represent numerical values with arbitrary accuracy
- *Disadvantages:* Codes need version of Mathematica. Viewing notebooks also requires some parts of Mathematica be installed.

Python

- Python is an interpreted language that shares many features with C and Matlab. These types of languages are often called “scripting” languages
- It has common approach to “quick” solutions to problems and has a very large community of developers (open source)
- No variables need be declared and often program development can be fast in this language.
- Program are created with a text editor.
- The name comes from Monty Python’s Flying Circus.

Parallel computing

- One of the methods available to carrying out large computations. Basic idea is to break problem into smaller parts that do not depend directly on each other and can be evaluated simultaneously on separate computers.
- Linux based PCs make this approach relatively inexpensive.
- Not quite “off-the-shelf” yet but progressing rapidly (e.g., later in course we will look at parallel Matlab).

General approach to any computational problem:

- Statement of problem: The clearer this can be done, the easier the development and implementation
- Solution Algorithm: Exactly how will the problem be solved.
- Implementation: Breaking the algorithm into manageable pieces that can be coded into the language of choice, and putting all the pieces together to solve the problem.
- Verification: Checking that the implementation solves the original problem. Often this is the most difficult step because you don't know what the “correct'' answer is (reason for program in the first place).

Scientific Computing

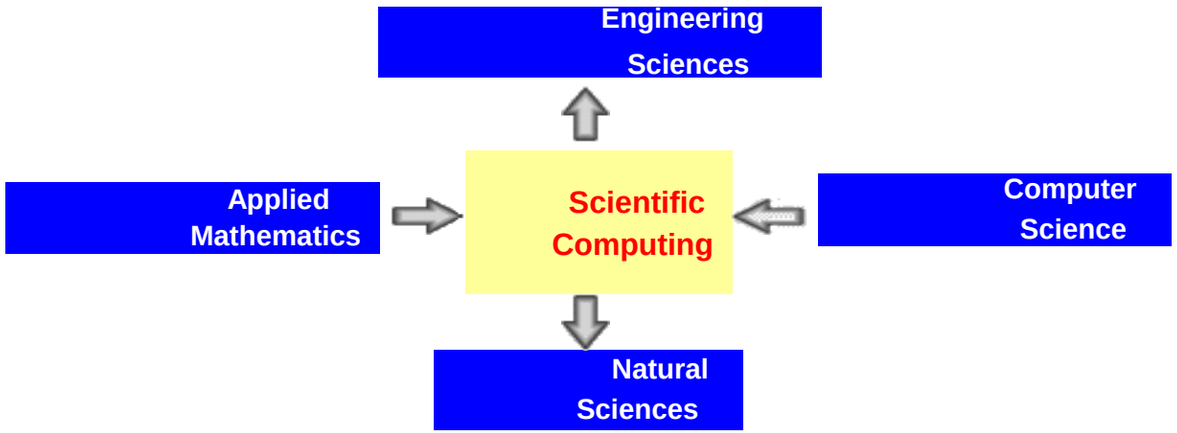
What is Program

- A Set of Instructions
- Data Structures + Algorithms
- Data Structure = A Container stores Data
- Algorithm = Logic + Control

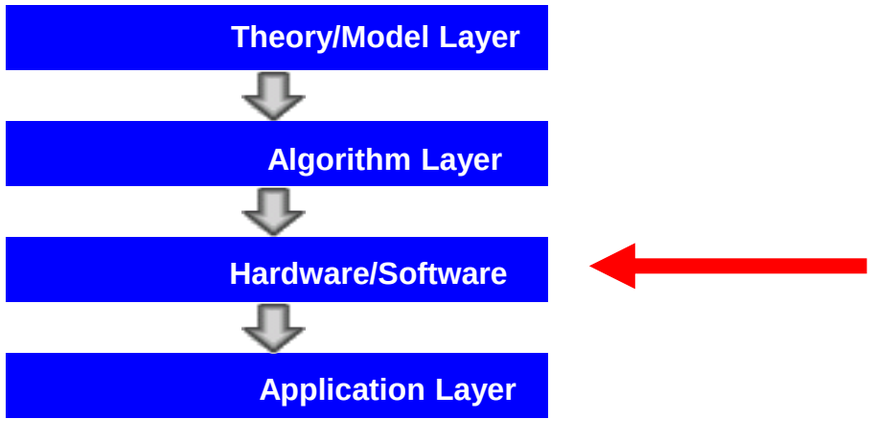
What is Scientific Computing?

- Short Version
  - To use high-performance computing (HPC) facilities to solve real scientific problems.
- Long Version, from *Wikipedia.com*
  - Scientific computing (or computational science) is the field of study concerned with constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific and engineering problems. In practical use, it is typically the application of computer simulation and other forms of computation to problems in various scientific disciplines.

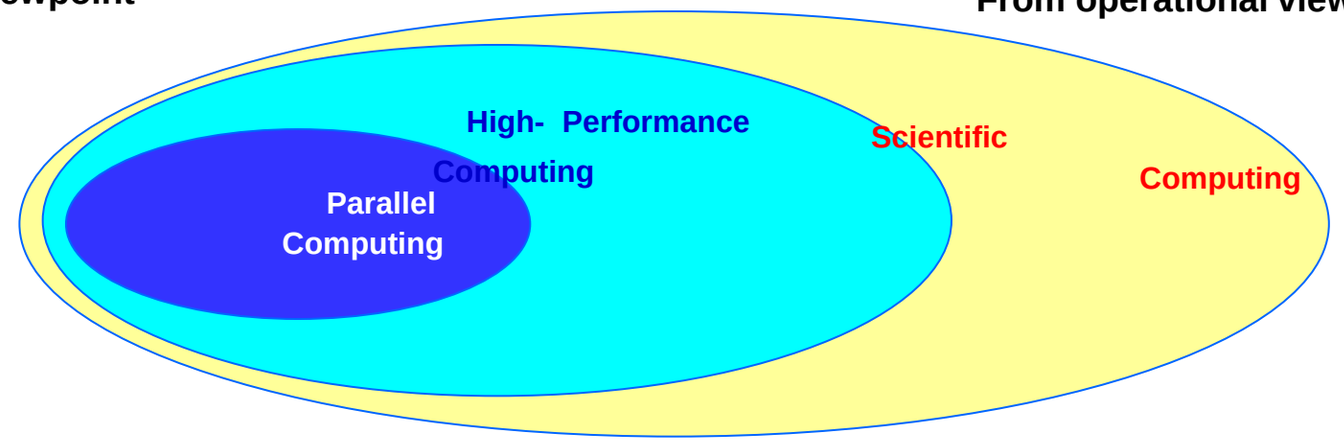
What is Scientific Computing?



From scientific discipline viewpoint



From operational viewpoint



From Computing Perspective

What is HPC (High Performance Computing)

- *Computing resources which provide more than an order of magnitude more computing power than current top-end workstations or desktops* – **generic, widely accepted**.
- HPC ingredients:
  - large capability computers (fast CPUs)
  - massive memory
  - enormous (fast & large) data storage
  - highest capacity communication networks (Myrinet, 10 GigE, InfiniBand, etc.)
  - specifically parallelized codes (MPI, OpenMP)
  - visualization

Why HPC?

- What are the three-dimensional structures of all of the proteins encoded by an organism's genome and how does structure influence function, both spatially and temporally?
- What patterns of emergent behavior occur in models of very large societies?
- How do massive stars explode and produce the heaviest elements in the periodic table?
- What sort of abrupt transitions can occur in Earth's climate and ecosystem structure?
- How do these occur and under what circumstances? If we could design catalysts atom-by-atom, could we transform industrial synthesis?
- What strategies might be developed to optimize management of complex infrastructure systems?
- What kind of language processing can occur in large assemblages of neurons?
- Can we enable integrated planning and response to natural and man-made disasters that prevent or minimize the loss of life and property?

“Scientific Computing and Visualization”

SCV - Scientific Computing and Visualization

- Scientific Computing and Visualization
- Computing
  - high-performance, parallel computers (SCF)
  - support
    - parallelization
    - optimization
    - tutorials
- Visualization
  - support to create graphics, from publication figures to interactive 3D displays and movies
  - 15’x8’ tiled display for 3D visualization
  - tutorials

Usage Notes

- These slides were gathered from the presentations published on the internet. I would like to thank who prepared slides and documents.
- Also, these slides are made publicly available on the web for anyone to use
- If you choose to use them, I ask that you alert me of any mistakes which were made and allow me the option of incorporating such changes (with an acknowledgment) in my set of slides.

Sincerely,

Dr. Cahit Karakuş

cahitkarakus@esenyurt.edu.tr