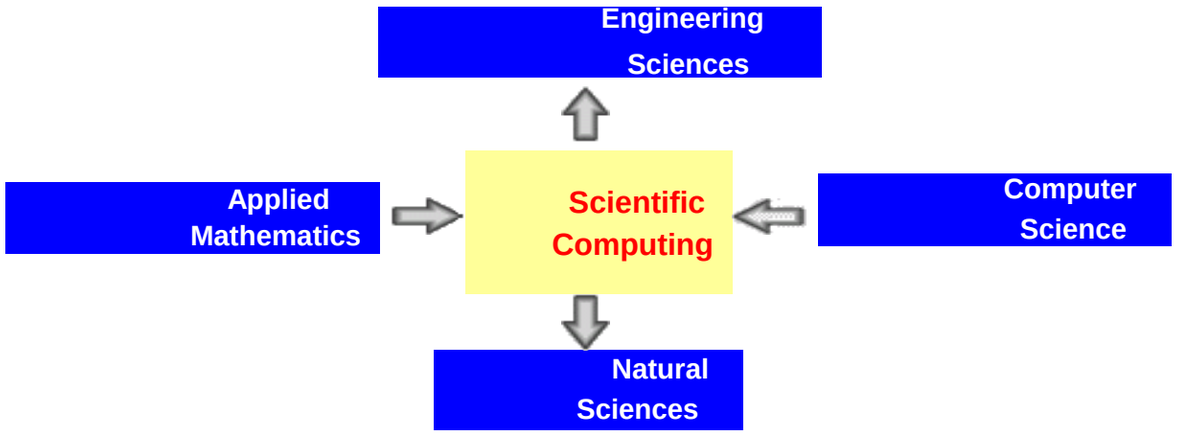
“Scientific Computing”

Dr. Cahit Karakuş, 2021

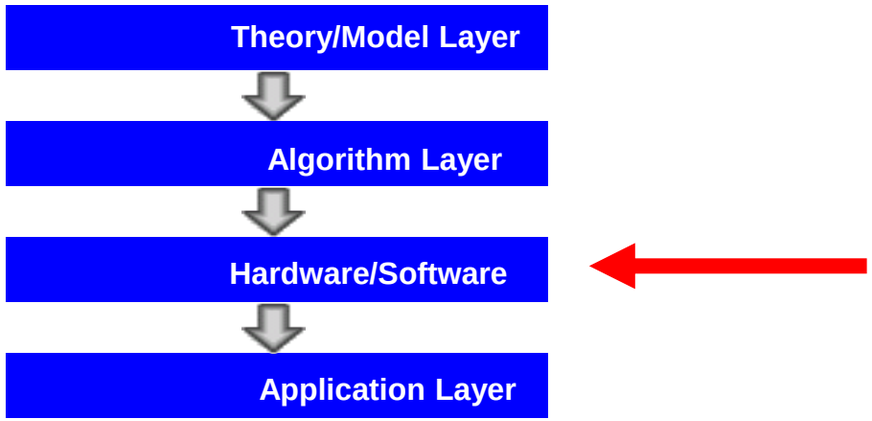
What is Scientific Computing?

- 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.

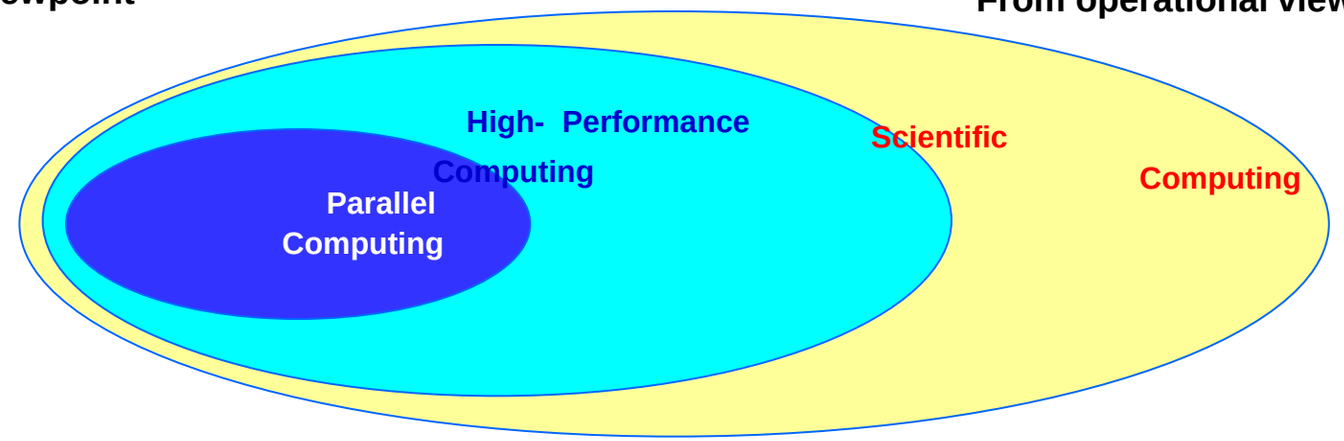
What is Scientific Computing?



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?

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.

Computer Science vs. Management Information Science

- Bilgisayar bilimi yoğu programlamadır.
- Involves
  - system architecture
  - software engineering
  - application programming
  - hardware
  - theory

- Bilgi Teknolojisi, organizasyonla ilgili uygulamalardır
  - business related
  - organizational automation

Lisans Düzeyi

- Software Development
- Software Engineering
- Database Administration
- Database Programming
- Internet Engineering
- Web Development
- Systems Administration
- Network Administration
- Information Technology

Research Project Areas

- Yapay zeka
- Biyoinformatik ve biyoteknolojiye uygulama
- küme hesaplama
- Bilgisayar grafikleri ve oyun tasarımı
- Veritabanı ve veri madenciliği
- Görüntü işleme
- internet uygulaması
- Multimedya ve programlama dilleri
- Ağ, İnternet Mühendisliği ve dağıtılmış sensör ağları
- Yazılım Mühendisliği
- Paralel ve dağıtık sistemler, Görselleştirme
- Web tabanlı eğitim

BS Degree Requirements - 1

- Foundation Courses
- Introduction to Computer Science
  - Discrete Structures for Computer Science
  - Computer Science I: Programming & Problem Solving
  - Computer Science II: Data Structures & Abstraction
  - Computer Architecture

- Mathematics Courses
  - Calculus I
  - Calculus II
  - Linear Algebra

BS Degree Requirements - 2

- Core Upper Division Courses
  - Operating Systems
  - Structure of Programming Languages
  - Social & Ethical Issues
  - Algorithms
  - Networks
  - Databases
  - Software Engineering
  - Project

- What is CS “really”?
- Is it like something else?
  - math?
  - electrical engineering?
- Is it “sui generis”

Is CS “sui generis”?

- "Bilgisayar biliminin diğer birçok konu ile o kadar yakın ilişkileri vardır ki, onu kendi başına bir şey olarak görmek zordur." Marvin Minsky“
- Bilgisayar bilimi, bilinen bilimlerden o kadar farklıdır ki, bilimler arasında yeni bir tür olarak görülmesi gerekir." Juris Hartmanis

Fundamental Principle of “What Is” Questions

- There are no sharp boundaries in nature
  - only continua & spectra
- **We** “carve nature at joints” (Plato) of our own devising (Kant)
- ∴ There may be no good answer beyond:
  - “CS is what CS’ists do!”
  - But: What do they do?

Newell, Perlis, & Simon 1967
Newell & Simon 1976
Simon 1969/1996

- CS = the science of computers
  - not a “natural science”, but:
    - a “science of the artificial”
  - “computers” includes:
    - hardware, algorithms, etc.
- So:
  - CS = the artificial **science** & engineering of **computers**
    … & surrounding phenomena

Knuth 1974

- 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!
- So:
  - CS = study of **algorithms**
    … & surrounding phenomena

Hartmanis 1992

- CS = study of **information**
  - how to represent info
  - how to process info
  - & the machines that do this

Shapiro 2001

- CS = natural science of **procedures**
  - including algorithms, …
    - & recipes (specifications; vague)
    - & non-halting (& interactive) procedures
    - & heuristics (incorrect O/P)

Brooks 1996

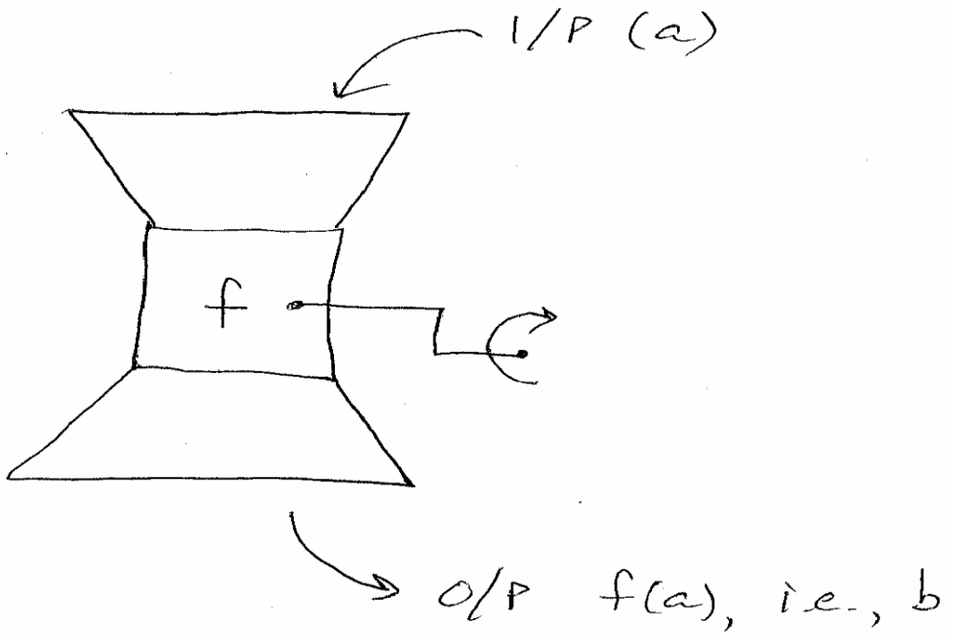
- CS ≠ science
  - not concerned with “discovery”
- CS = **engineering**
  - concerned with “making”:
    - physical computers
    - S/W systems

Denning 2010

- “Computing is a 4th great domain of science alongside the physical, life, and social sciences.”
- CS =
  - discovery (science)
  - & implementation (engineering)
  - of information processes

What Is Computation?

- **function** =
  - set of I/O pairs (relation)
  - same I/P $\Rightarrow$ same O/P
- "function machine":



- But: “function machine” ≠ function!
  - functions don’t “do” anything
  - function machine needs “gears”
    - i.e.) algorithm!
    - but not all functions are algorithmic (computable)!
  - function machine = …
    - computer!

Computable Functions

- A function **is computable** ≈
  - there is an **algorithm** that computes it
  - i.e.) that specifies **how …**
    - I/P can be transformed into O/P

Algorithm

- Algorithm for problem P ≈
  - **finite** procedure for solving P
    - finite # instructions
    - completable in finite time (?)
    - completable in finite # steps (?)
  - instructions **unambiguous** for executor
    - know how to do each instruction
    - know what to do next
  - must **halt** (?)
  - O/P must be **correct** (?)

4 Great Insights of CS

1. Bacon/Leibniz/Boole/Turing/Shannon/Morse:
   - **All information**
     about any computable problem
     can be represented
     using only <u>2 nouns</u>: 0, 1

2. Turing's Insight:

– Every algorithm can be expressed in a language for a **Turing machine**:

- arbitrarily long tape divided into squares
- read / write head
- **only 2 verbs** (= basic instructions):
  - move(*dir*) (where *dir* = L, R)
  - print(*sym*) (where *sym* = 0, 1, nil)

3. Boehm & Jacopini's insight:

- structured programming
- only need 3 grammar rules:
  - sequence (first do this**;** then do that)
  - selection (**if** P, **then** do this **else** do that)
  - repetition (**while** P **do** this)
- recursively, "this" & "that" can be:
  - any basic instruction
  - any complex instruction created by grammar rules
- optional:
  - exit
  - named procedures
  - recursion

4. Church-Turing Thesis:

- **An algorithm is$_{def}$**
  anything equivalent to a Turing-machine program

- TM = Post production system = lambda calculus
  = Markov algorithm = recursive functions
  = register machines, etc.

- There are non-computable functions
  - e.g.) Halting Problem

So, what *is* CS?

- CS tries to answer these questions:

1. **What is computation?**
   - More generally, what is a procedure?
2. Which functions are (or are not) …
   - **computable?**
   - **efficiently** computable?
     - theoretical CS
3. How can they be computed?
   - what specific **algorithms** do the job?
   - what **machines** are needed?
     - other branches of CS

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.

The computer is incredibly fast, accurate, and stupid. Man is unbelievably slow, inaccurate, and brilliant.

The marriage of the two is a force beyond calculation. -Leo Cherne

“Everybody in the country should learn how to program a computer... Because it teaches you how to think.” -Steve Jobs

Computer Science is Infrastructure and Networks: Keep computer systems up and running? Invent new ways for technologies to connect? Computing is also the connections, the networks between computers.

Computer Science is Computer Forensics and Cyber Security: Solve crime? Keep us safe? Secure information?

Computer Science is Design: Make models? Design cars, houses, fashion, anything?

This is the skill set of people who accomplish a lot through computer science. These skills are essential to working on technical teams and to invent new ways of doing things with computers.

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.

Paths to Careers
What skills does CS require?

Computer Science might be right for you if you have…

- Curiosity and imagination
- Flexible, creative thinking
- Work ethic – make an effort in math and science
- Communication skills in order to tackle challenges with others

Computer science

- Computer science is vocational
- The right way to think about computer science in our education system is that it is foundational

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

Paths to Careers: *How do people get started?*

Keep taking Science and Math
Learn Computer Science

A first high school CS course might have you studying…

- Human Computer Interactions
- Problem Solving
- Web Design
- Introduction to Programming
- Robotics
- Computing Applications

Discuss with the host in advance what computer science offerings are available and add a second slide following this one.

Computers

- Computers process information by computing new results and answering queries.
- Computers input and output information
- Computers are general purpose because they can perform many different tasks
- Computers are programmable so they can remember sequences of operations
- a general purpose, programmable, information processor with input and output

Embedded computers and robots

- Machines with full-fledged computers inside
  - Washing machines, airplanes, ATMs, etc.
- Such machines require highly reliable, predictable computer programs
- All physical mechanisms controlled by computers are *robotic devices*
  - Restrict definition to machines that are general purpose and programmable
    - Robotic arm or cart

What can computers do – today?

- Business productivity managers
- Personal information managers
- Spreadsheets
- Database software
- Desktop publishing
- Multimedia encyclopedias
- Simulate the physical world
- Produce a music video

What might computers do – tomorrow?

- Diagnose diseases
  - MYCIN captures medical knowledge in rules that allowed a computer to identify an ailment based on symptoms
- Control robots that walk, talk, and learn
  - CMU created a program that drove a van from Pittsburgh to D.C. using cameras for eyes
- Compose music and create art

How do computers solve problems?

- Humans deconstruct problems into small operations that a computer can carry out
  - Writing an *algorithm*
- Solve a problem by computer requires
  - State the problem clearly in a *problem statement*
  - Solve the problem with an *algorithm* that gives clear instructions
  - Use a *computing agent* to carry out the instructions

Stating the problem clearly

- Describes *what* to do, not *how* to do it
  - How do I get from Timonium Campus to the Beltway?
- Solve general classes of problems
  - How do I get from point *A* in Timonium to point *B*?
  - What is the square root of *y*?

Specifying a problem

- Clear problem statement is called the *specification*
  - What information can we use as input
  - What the output, or solution, to our problem should look like
- Specification for the square root problem
  - Input: A positive number $y > 0$
  - Output: A positive number x such that $x^2 = y$
- Make sure specification is not ambiguous

Solving the problem using an Algorithm

- Algorithm – a clear sequence of instructions for performing a task
  - a well-ordered sequence
  - of well-defined,
  - feasible operations
  - that takes finite time to carry out

What is computer science?

- The study of computers
- The study of algorithmic processes including their
  - Theory
  - Analysis
  - Design
  - Efficiency
  - Implementation
  - Application

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?

Who invented computers?

- Computer science has roots in two fields
  - Mathematics
    - Alan Turing and the Turing machine (1930s)
    - Developed theories with paper and pencil about how to perform computations by hand
  - Engineering
    - John von Neumann and the von Neumann machine (1940s)
    - Showed how to build physical computers out of electronic circuitry

Mathematical Roots

- Leibniz's Dream (1600s)
  - Can we find a universal language for mathematical algorithms that will let us describe and solve any problem?
    - Reduce all reasoning to a fixed set of basic rules
    - Determine truth or falsity of sentences by fixed rules for manipulating sentences
- George Boole (1800s)
  - Introduces binary notation of calculation
    - Computers use binary system for logic and arithmetic

More on Theory

- David Hilbert (1928)
  - Challenges the mathematical community to find an infallible, mechanical method for constructing and checking truth of mathematical statements
    - Interested in an algorithm
- Alonzo Church, Alan Turing, and Kurt Gödel construct arguments that there is no solution to Hilbert's Challenge
  - Turing builds a conceptual computer for his argument

The Turing Machine and the Church-Turing Thesis

- Turing Machine
  - Machine with a finite set of rules and an infinite amount of “scratch paper” for computation
    - No one has designed a physical computer that can do more than a Turing machine
  - Machine could not solve Hilbert’s problem
- Church-Turning Thesis
  - The Turing Machine captures what we mean by computational systems
  - Is as powerful an any other mechanical computing agent

Engineering Roots

- First step development of calculators
  - Abacus – developed 5000 years ago in the Middle East
  - Pascaline – first mechanical calculator using gears for calculation (1642)
  - Charles Babbage's Difference Engine – conceptual design that used hundreds of gears to compute mathematical functions (1820s)

Electronic Circuits

- Telegraph – uses electricity to convey letters and transmit information quickly (1844)
- Hollerith Tabulating Machine – Uses electricity and punch cards to calculate the US census (1890)
- Z2 – used circuitry to compute arithmetic operations (1930s)

Programmed Devices

- Jacquard Loom – weaves cloth using a pattern specified using punch cards (1801)
- The Analytic Engine – conceptual design for a machine consisting of a Mill, Store, Printer, and Readers
  - Led Ada Lovelace to define programming concepts such as the subroutine
- ENIAC – one of the first programmable electronic computers (1945)
  - Programmed by routing cables and flipping switches

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.

Coding

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.

57

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)

59

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.

60

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.

61

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

62

İLK PROGRAM

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


63

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

65

VERİ TİPLERİ

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

66

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

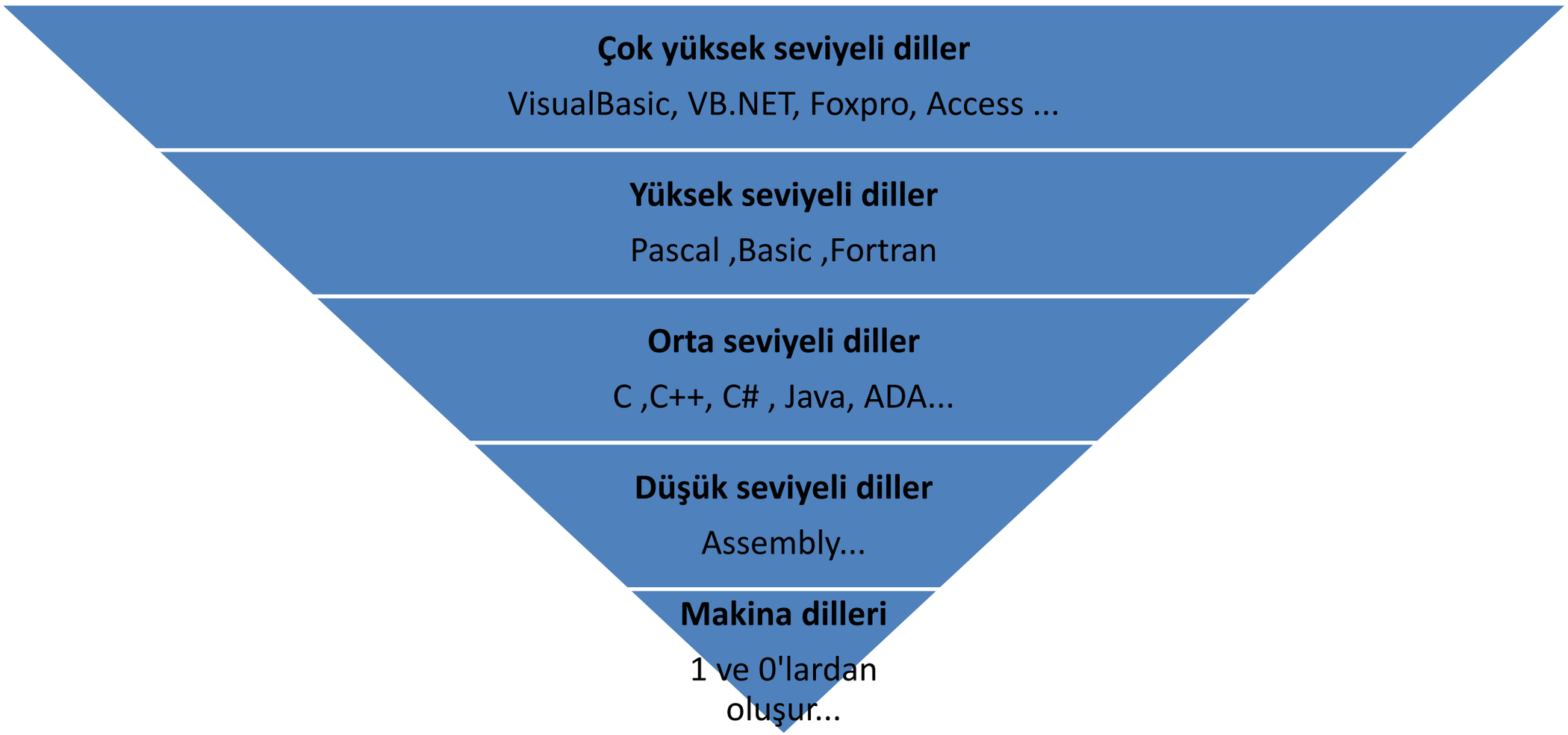
O-notation	Adjective Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n \log n)$	n log n
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential
$O(10^n)$	Exponential
$O(2^{2^n})$	Doubly exponential

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.

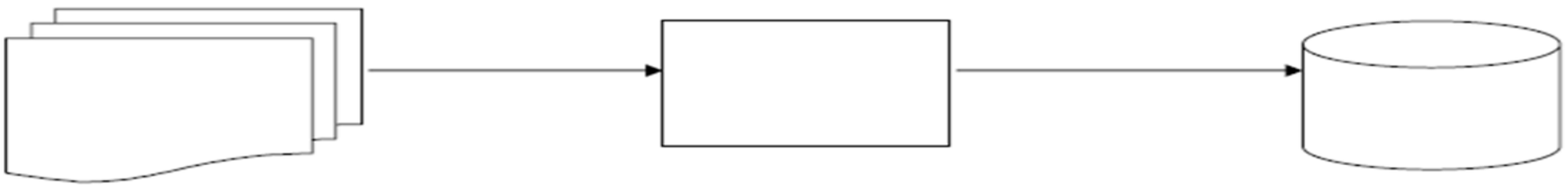
Makine dili dışındaki tüm diller semboliktir ve makine diline dönüştürülmesi gerekir.

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

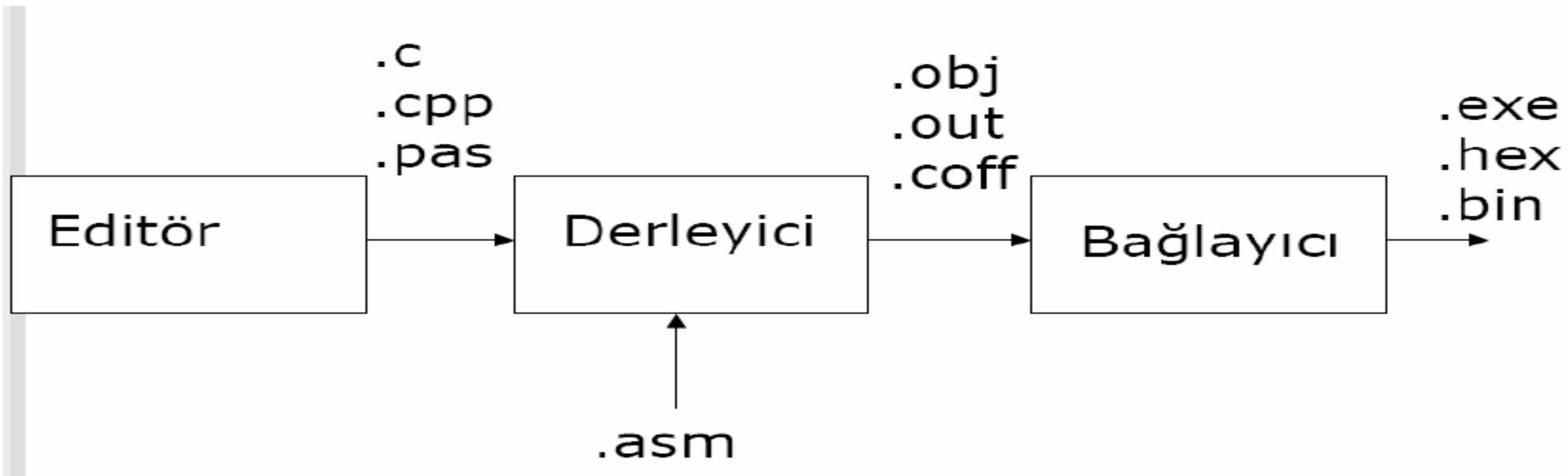
Programlama Dili Nedir?

- 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



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

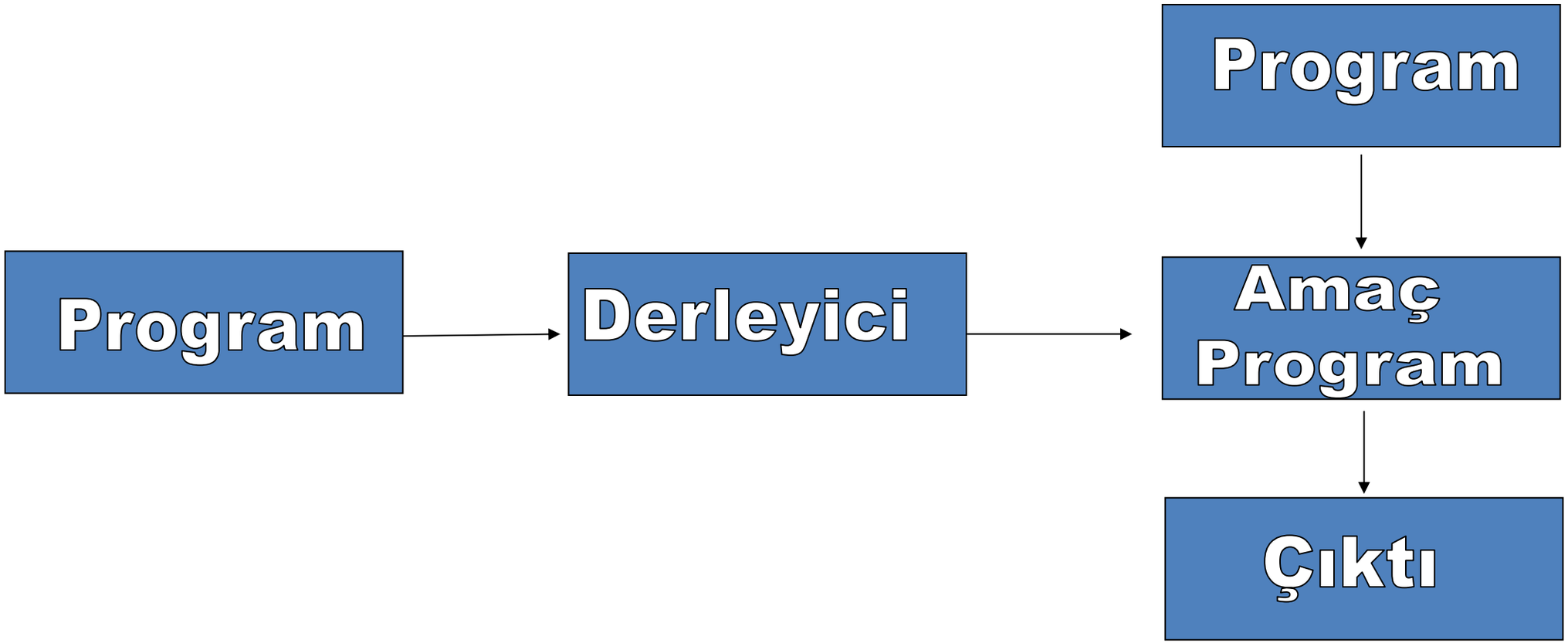
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	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	

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.

- 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.

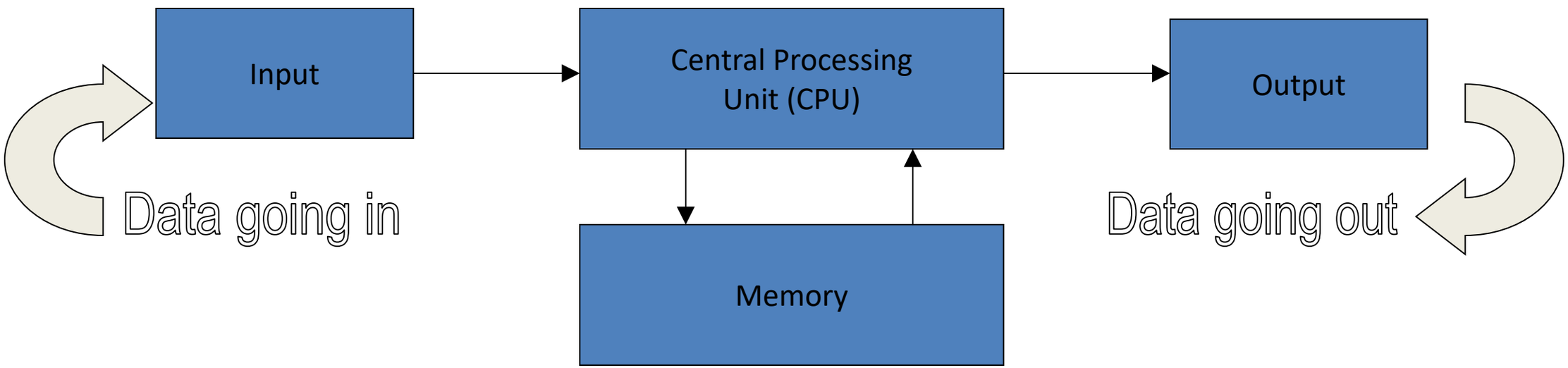
“Scientific Computing and Visualization”

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

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



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”

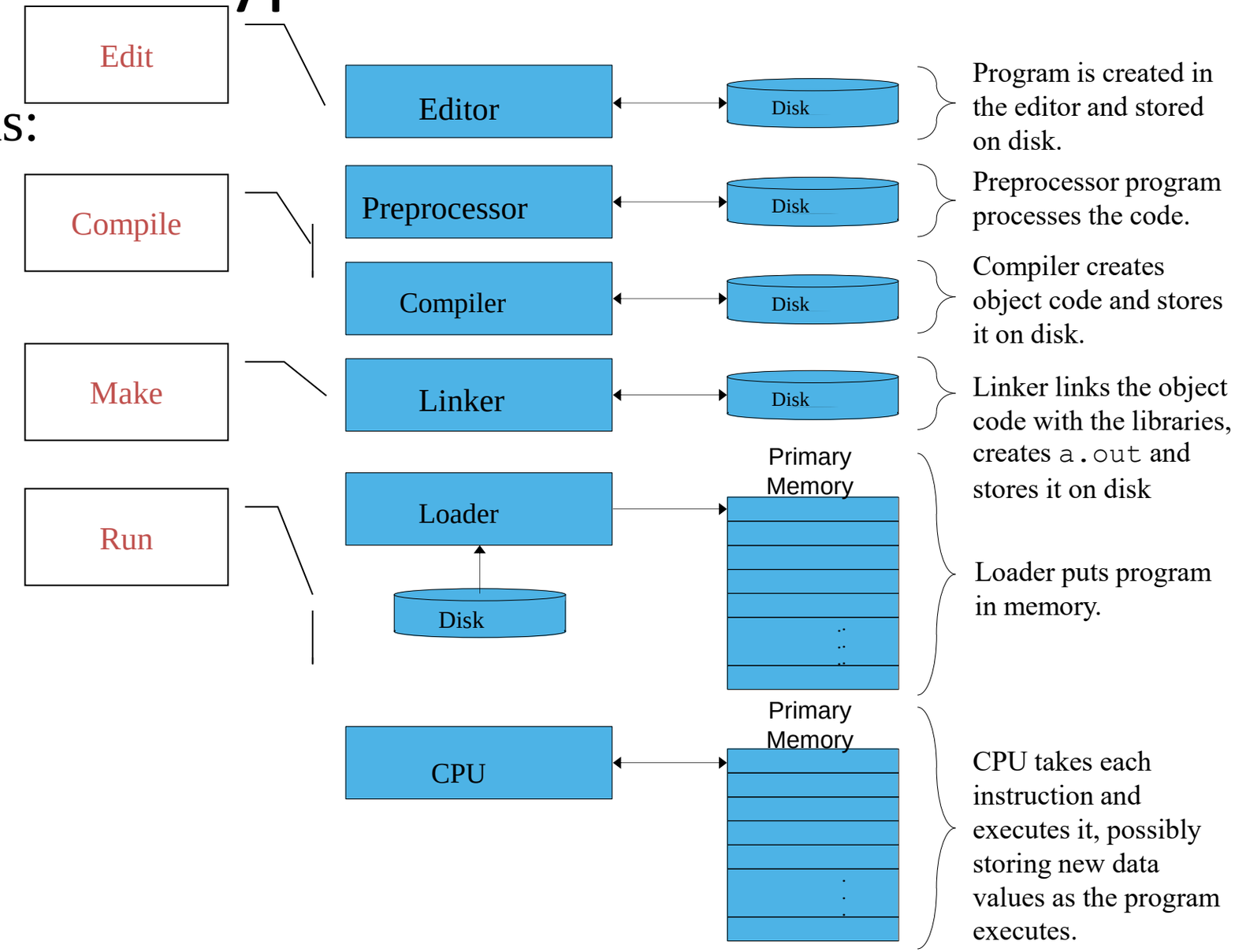
Basics of a Typical C++ Environment

- 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).

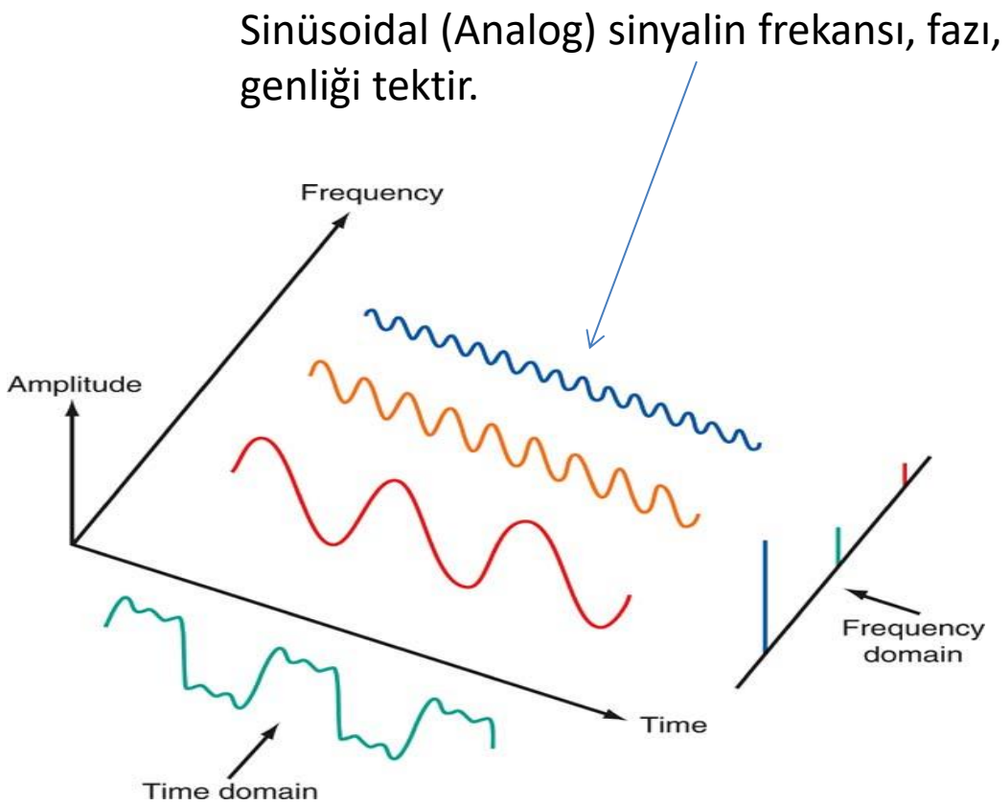
Fourier Theory

Fourier Theory

- Özellikle sinüs dışı dalga yaklaşımı için bir iletişim devresine veya sistemine ait sinyallerin karakteristiklerini ve performansını belirlemek için kullanılan bir yöntem Fourier analizidir.
- Fourier teorisi, sinüzoidal olmayan sinyalde bir dalga formunun, harmonik olarak ilişkili bireysel sinüs dalgası veya kosinüs dalgası (sinüsoidal sinyal) bileşenlerine ayrılabileceğini belirtir.
- Bir kare dalga bu fenomenin klasik bir örneğidir.

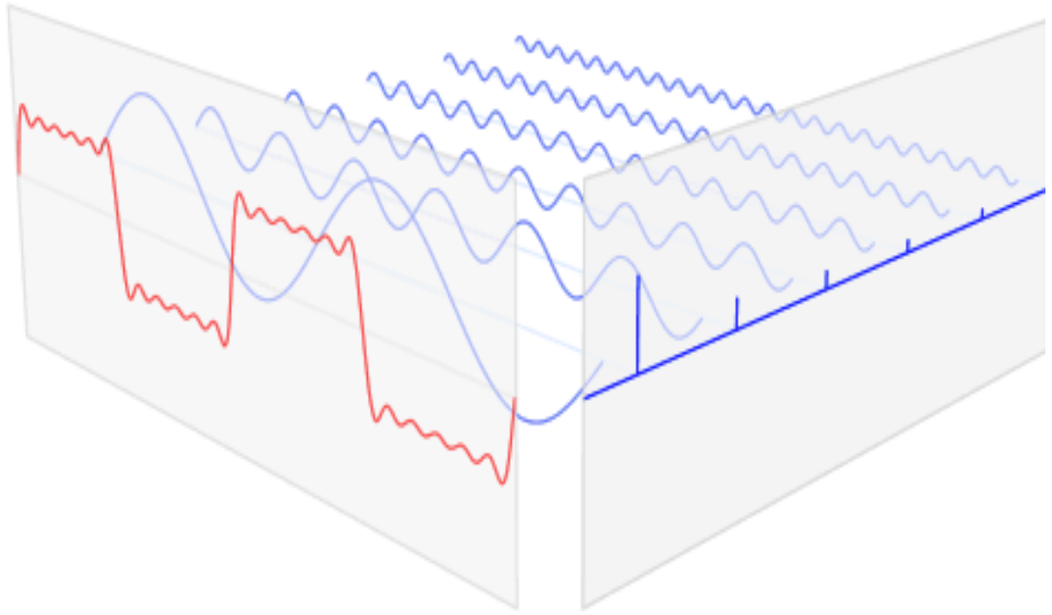
Temel İçerik:

- Fourier analizi, bir sinyal sonsuz sayıda harmonik sinüsoidal sinyallerden oluştuğunu belirtir.
- Fourier analizi, yalnızca karmaşık bir sinyaldeki sinüs dalgası bileşenlerini (frekans, genlik, faz) değil, aynı zamanda bir sinyalin bant genişliğini belirlememizi sağlar.



Fourier dönüşümü

- Fourier dönüşümü yaygın olarak zaman spektrumundaki bir sinyali bir frekans spektrumuna dönüştürmek için kullanılır. Zaman spektrumlarına sinyal örnekleri olarak ses dalgaları, elektriksel sinyaller, mekanik titreşimler vb. Aşağıdaki şekil açıkça görülebileceği gibi, Kare dalga farklı frekanslara sahip bir dalgaya benziyor. Aslında birden çok dalgaya benziyor.
- Fourier Dönüşümü doğrusal olmayan her fonksiyonun (sonsuz) sinüs dalgalarının bir toplamı olarak temsil edilebileceği gerçeğinden yararlanır. Alttaki şekilde bu, bir basamak fonksiyonu çok sayıda sinüs dalgası tarafından simüle edildiği için gösterilmiştir.



Fourier Dönüşümü

- Evrende gözlediğiniz tüm sinyal formları farklı frekans ve genliklere sahip sinüs fonksiyonlarının toplamından ibarettir!
- Sinyaller, bir dalga formu aracılığıyla tanımlanabilir - zaman, mekan veya başka bir değişkenin fonksiyonu. Örneğin, ses dalgaları, elektromanyetik alanlar, radyodan dinlediğiniz müzik, hisse senetlerini zamana göre fiyatı, nefesinizin sıklığı vb.
- Fourier Dönüşümü, bize bu dalga formlarını doğrudan görüntülemenin benzersiz ve güçlü bir yolunu sunduğu için önemli bir rol oynamaktadır.
- Fourier Transform, sinyal analizi, görüntü analizi, görüntü filtreleme, görüntü rekonstrüksiyonu ve görüntü sıkıştırması gibi çok çeşitli uygulamalarda kullanılır.
- Fourier dönüşümü fizik ve mühendislik alanındaki birçok uygulama ile matematiksel bir dönüşümdür. Fourier serileri denilen trigonometrik serileri kullanarak kısmi diferansiyel denklemleri içeren birçok önemli problemi çözebiliriz.

Fourier Transform

Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \qquad \text{Analysis}$$

Inverse Fourier Transform:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega \qquad \text{Synthesis}$$

Continuous-Time Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt \qquad f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(j\omega)e^{j\omega t}d\omega$$

Discrete-Time Fourier Transform(DTFT):

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} \qquad x[n] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\omega})e^{j\omega n}d\omega$$

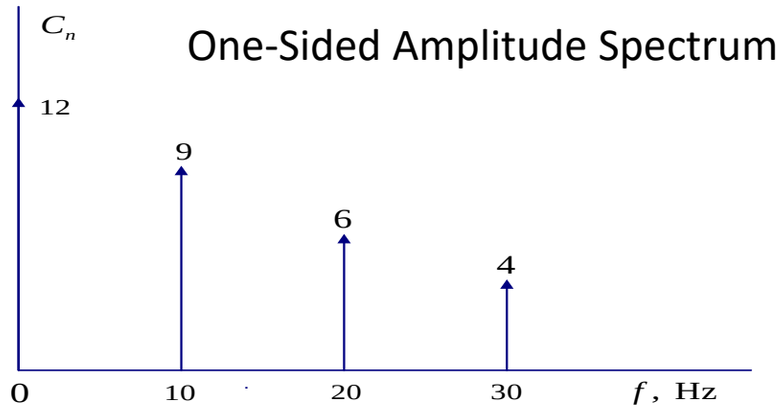
- Zamanla genliği değişen analog sinyallerindeki frekansların belirlenmeside ve frekansa göre genliklerinin ayrıştırma Fourier dönüşümü.
- Frekansların bileşenlerinden zaman domeninde analog sinyal elde edilmesi sentez. Frekanslara göre birleştirme.
- Peryodik Sinyallarin Fourier Dönüşümü daha doğru hesaplanır.
  Bir analog sinyalin Fourier dönüşümü komplks sayılar içerir.

Example: frequencies and plot the one-sided amplitude spectrum.

- X(t)=Acos(wt+θ), x(t): Analog sinyal, eğer frekans tek ise sinüsoidal sinyal olarak adlandırılır.
- A: Genlik
- w: Açısal frekans (Dalga biçiminde yayılan bir sinyal söz konusu)
- w=2πf , f=1/T; Burada f: frekans (Hz), T: peryod (saniye)
- Θ: faz açılarıdır, radyan cinsinden verilir (derecede verilebilir).

The frequencies are
0 (dc)
10 Hz
20 Hz
30 Hz

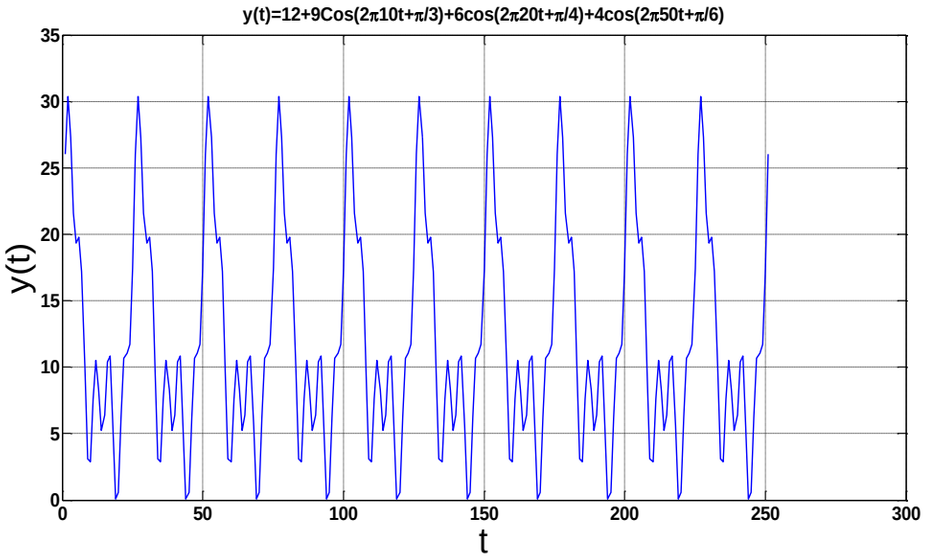
$$x(t) = 12 + 9\cos(2\pi \times 10t + \pi/3) + 6\cos(2\pi \times 20t - \pi/6) + 4\cos(2\pi \times 30t + \pi/4)$$



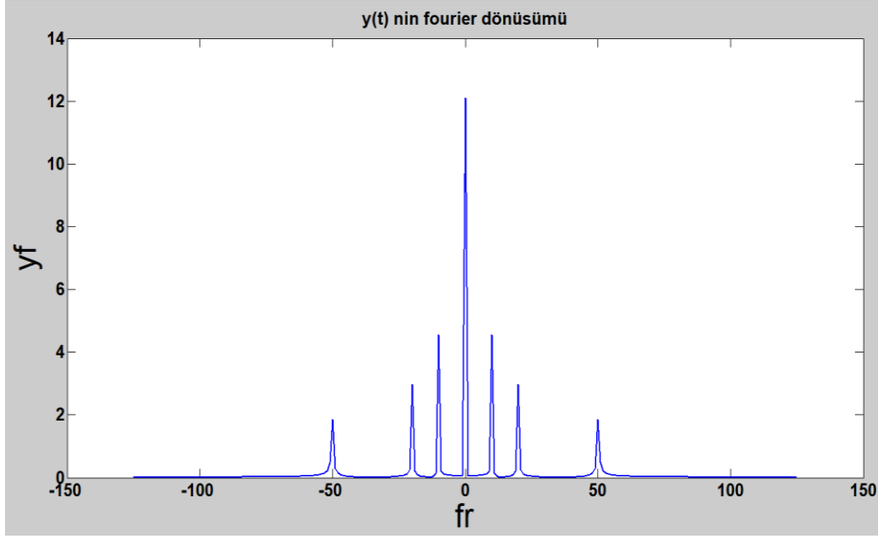
Örnek

```
clear all;  close all
f1=0; f2=10;f3=20;f4=50;
A1=12;,A2=9;A3=6;, A4=4;
faz1=0; faz2=pi/3;faz3=pi/4;faz4=pi/6;

fs=5*f4
p=fs/f2
Ts=1/fs
N=round(1/Ts)

for i=1:N+1
   t(i)=0+(i-1)*Ts;
   fr(i)=-round(N/2)+i-1;
end

for i=1:N+1
   y1(i)=A1;
   y2(i)=A2*sin(2*pi*f2*t(i)+pi/3);
   y3(i)=A3*sin(2*pi*f3*t(i)+pi/4);
   y4(i)=A4*sin(2*pi*f4*t(i)+pi/6);
   y(i)=y1(i)+y2(i)+y3(i)+y4(i);
end
```

```
figure, plot(y,'LineWidth', 2, 'MarkerSize', 10)
grid on;
title('y(t)=12+9Cos(2\pi10t+\pi/3)+6cos(2\pi20t+\pi/4)+4cos(2\pi50t+\pi/6)', 'FontSize', 20, 'Color', 'k', 'FontWeight', 'bold');
xlabel('t','FontSize',36)
ylabel('y(t)','FontSize',36)
set(gca,'FontSize',20, 'FontWeight','bold');
```




```
S0=fft(y);
S1=abs(S0)/N;
figure, plot(fr,fftshift(S1),'LineWidth', 2, 'MarkerSize', 10)
title('y(t) nin fourier dönüsümü', 'FontSize', 20, 'Color', 'k', 'FontWeight', 'bold');
xlabel('fr','FontSize',36)
ylabel('yf','FontSize',36)
set(gca,'FontSize',20, 'FontWeight','bold');
```



2D Fourier transform

Definition

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy,$$
$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

where u and v are spatial frequencies.

Also will write FT pairs as $f(x,y) \Leftrightarrow F(u,v)$.

- $F(u,v)$ is complex in general,

$$F(u,v) = F_{\mathrm{R}}(u,v) + jF_{\mathrm{I}}(u,v)$$

- $|F(u,v)|$ is the magnitude spectrum
- $\arctan(F_{\mathrm{I}}(u,v)/F_{\mathrm{R}}(u,v))$ is the phase angle spectrum.
- Conjugacy: $f^*(x,y) \Leftrightarrow F(-u,-v)$
- Symmetry: $f(x,y)$ is even if $f(x,y) = f(-x,-y)$

2D Fourier Transform

- The 2D Fourier Transform equation is very similar to the 1D

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\frac{u\,x}{M}+\frac{v\,y}{N})}$$

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(x,y)e^{i2\pi(\frac{u\,x}{M}+\frac{v\,y}{N})}$$

- The 1/MN term can be applied to either function (but not both) and is used for normalization

- In MATLAB the call for a 2D Fourier transform is

```
>>F = fft2(f)
>>f = ifft2(F)
```

Laplace Dönüşümü

Why use Laplace Transforms?

- Laplace dönüşümü, Fransız matematikçi tarafından (1749-1827) geliştirildi ve geçen yüzyılda mühendislik problemlerine geniş ölçüde uyarlandı.
- Faydası, diferansiyel denklemleri daha kolay çözülen cebirsel biçimlere dönüştürme yeteneğinde yatmaktadır. Belli alanlarda sistemlerle uğraşmak için bir mühendislik “dili” biçimi olarak çok yaygın hale gelmiştir.
- Cebir kullanarak diferansiyel denkleme çözüm bulunur. Fourier Dönüşümü ile ilişkilidir, sistemlerin davranışlarını, yörüngelerini karakterize etmenin kolay yolunu sunar.
- Diferansiyel denklem çözümlerinde dönüşüme ya da evrişime gerek yoktur. Sistemde birden çok işlem için kullanışlıdır.
- Fourier dönüşümü, sadece sinüs ve kosinüs temel işlevleri kullanılarak sentezlenebilen sinyalleri analiz etmek için yeterlidir.
- Sentez: Bileşenlerden bütünleşik bir sistem oluşturma
- Analiz: Bütünleşik bir sistemi bileşenlerine ayrıştırma
- Sinyalin üstel bileşenleri olduğunda, örneğin, zaman içinde üssel olarak değişkenlik gösteren sinüs dalgası için, fourier'in verdiği sadece yarım bilgidir. Özellikle de başlangıç koşullarına içeren kritik bilgiler kaybolur. Bu tür durumlarda laplace dönüşümü zorunludur.
- Aniden başlayan veya biten herhangi bir şeyi analiz etmek isterseniz, Laplace dönüşümünden büyük ölçüde faydalanırsınız. Bu geçişlere sistem cevabı gibi gerçek uygulamalarda çok karşılaşılır. Aksine, Laplace dönüşümü bir problemi büyük ölçüde basitleştirdiği asimetrik fenomenlerle uğraşırken Fourier dönüşümü çok verimsizdir. Laplace'ın çok önemli bir analitik aracı dönüştürmesini sağlar.

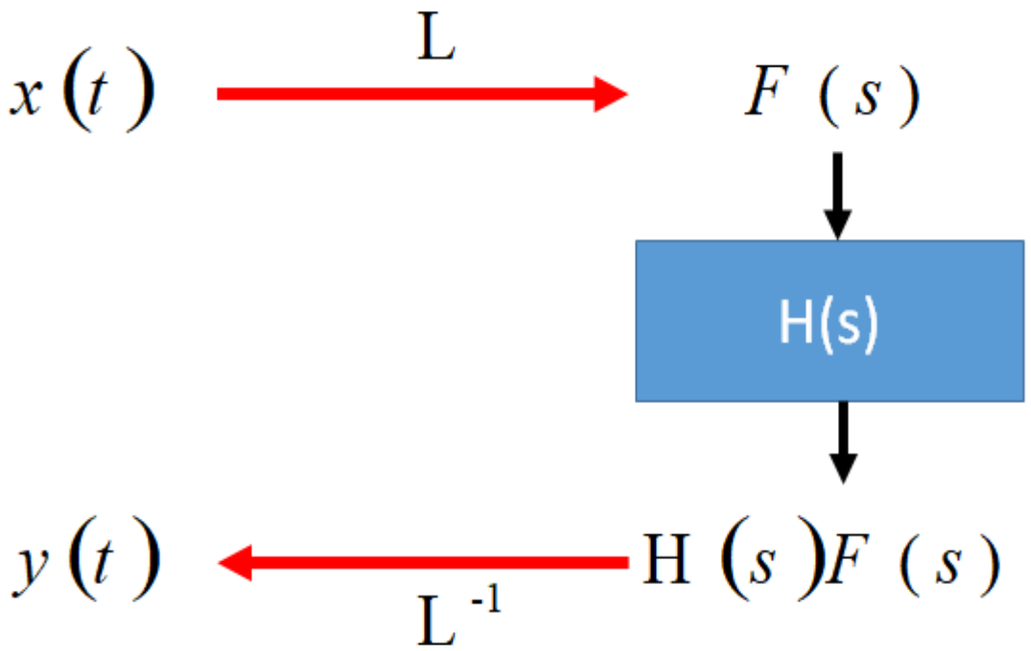
Laplace Dönüşümü

- Doğrusal adi diferansiyel denklemlerin çözümü için önemli bir analitik yöntemdir. Doğrusal olmayan ODE'lerde ilk önce doğrusallaştırılmalıdır. Laplace dönüşümleri önemli proses kontrol kavramları ve tekniklerinde önemli bir rol oynar.
- Sistemlerin, sadece bir frekans veya zaman alanı verileri yerine, sinusoidler ve üsteller gibi doğal bileşenler açısından analiz edilmesine yardımcı olur.
- Yoğun aktarım işlevini, kutuplar ve sıfırlar cinsinden (limit) tanımlanabildiği ve sezgisel olarak sistemin kararlılığını, aşılmasını veya gürültüsünü tahmin etmede yardımcı olan uygun bir alana (S-etki alanı) çevirir.
- Kutuplar: Paydayı sıfır yapan değişkenlerdir.
- Sıfırlar: Payı sıfır yapan değişkenlerdir.
- Fourier Dönüşümü, şifreli bir dönüşümü çok daha kolay bir çarpıma dönüştürürken, bir Laplace Dönüşümü, S-etki alanında basit bir polinom cebiri ile yoğun bir diferansiyel denklemin çözülmesine yardımcı olur.
- Yinelemeli filtrelerin kararlılığını analiz etmek için DSP'de temel olan Z-Dönüşümü, Laplace Dönüşümü'nün yakın akrabasıdır.
- Örnekler: Transfer fonksiyonları, Frekans tepkisi, Kontrol sistemi tasarımı, Kararlılık analizi

Transfer Function

- Sistem bir girişe ve bir çıkışa sahipse, çıkışın girişin bir fonksiyonu olarak tanımlandığı durumlarda, onu s etki alanına (laplace kullanarak) dönüştürebilir ve sistemi bir transfer fonksiyonu ile tanımlanabilir.
- X (s) 'in girdi olduğunu, Y (s)' nin çıkış olduğunu ve H (s) 'nin transfer fonksiyonu olarak bilinen şey olduğunu varsayalım. Sistem çıkışı Y (s) = H (s) * X (s) olarak tanımlanabilir.
- Transfer fonksiyonu Y (s) / X (s) ile eşit olacaktır.
- Transfer fonksiyonunun pay ve paydası faktörize edilirse, bir kutup-sıfır tanımı yapılabilir. Sistemin yüksek frekansları söndürmek için yapılmış bir elektrik devresi ise, düşük geçiş filtresi, sadece kutup-sıfır çizimi filtrenin belirli frekanslara nasıl tepki verdiği söylenebilir. Bu, laplace dönüşümünün sadece küçük bir uygulamasıdır, ancak birçok mühendislik dalında kullanışlı olabilen transfer fonksiyonları ile lineer sistemlerin tanımlanmasını sağlar.

Transfer Function

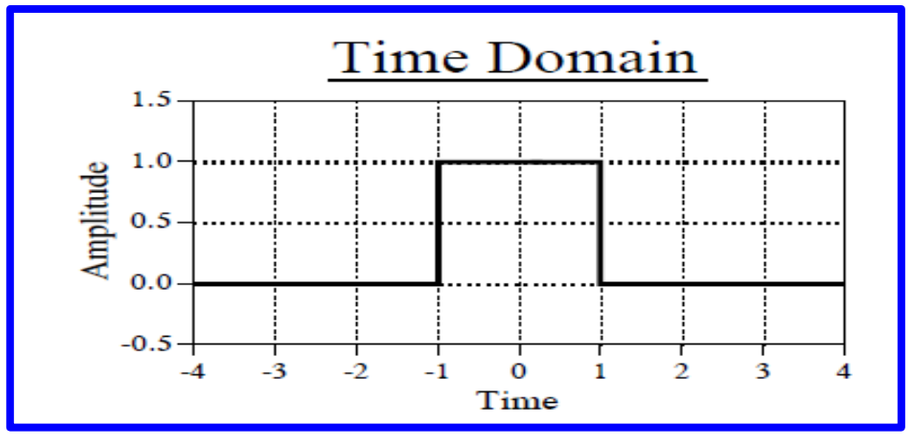


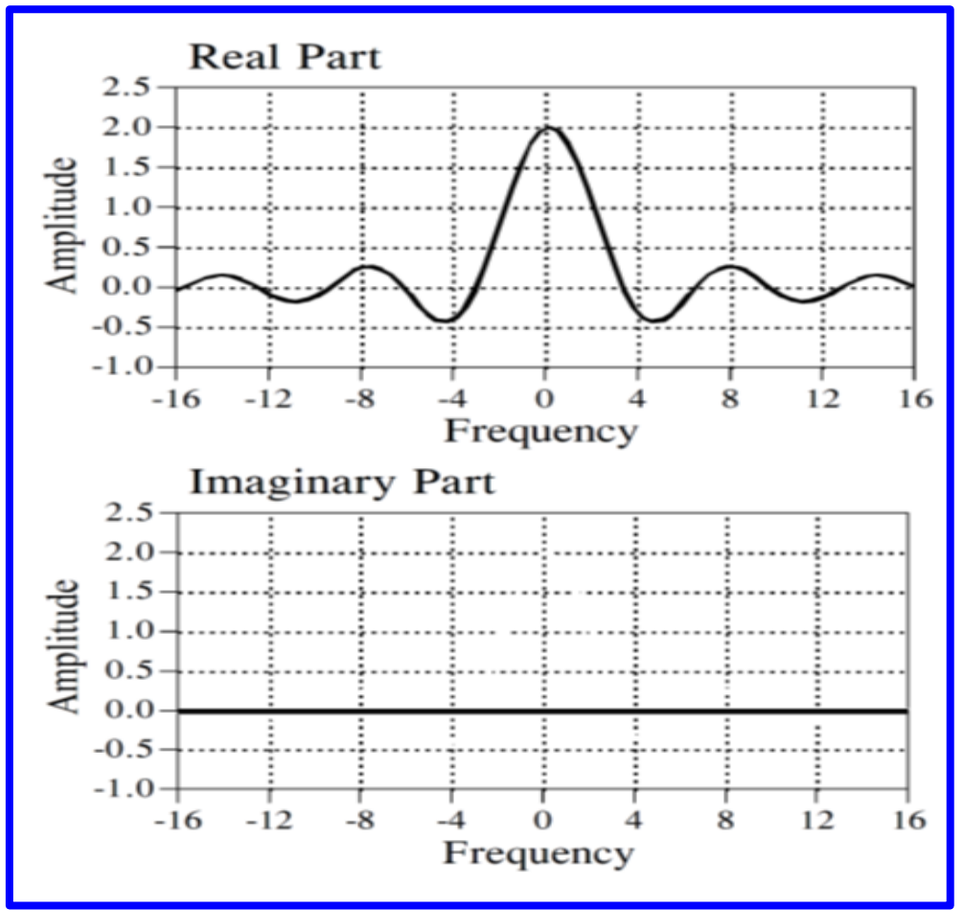
Transfer fonksiyonu: Kazanç, Kuvvetlendirme, Devre, Süzgeç …

Laplace transform can help us find y(t) easily

Fourier Transform

$$\mathrm{F}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} \cdot dt$$





$$\mathrm{f}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} \cdot d\omega$$

Laplace Dönüşümü

- Laplace dönüşümü ile sürekli-zaman Fourier dönüşümü arasındaki ilişki aşağıda gösterilmiştir.

- $s=jw$ için, $$X(s)=\int_{-\infty}^{\infty}x(t)e^{-st}dt \xrightarrow{s=jw} X(jw)=\int_{-\infty}^{\infty}x(t)e^{-jwt}dt$$

Dolayısı ile $$X(s)\Big|_{s=jw}=F\{x(t)\}$$

- $s=\sigma+jw$ için,

$$X(s)=\int_{-\infty}^{\infty}x(t)e^{-st}dt \xrightarrow{s=\sigma+jw} X(\sigma+jw)=\int_{-\infty}^{\infty}x(t)e^{-(\sigma+jw)t}dt$$

$$X(\sigma+jw)=\int_{-\infty}^{\infty}x(t)e^{-\sigma t}e^{-jwt}dt=\int_{-\infty}^{\infty}\left[x(t)e^{-\sigma t}\right]e^{-jwt}dt$$

Bu durumda eşitliğin sağ tarafının $x(t)e^{-\sigma t}$ 'nin Fourier dönüşümüne eşit olduğu görülür.

Constant Function

Constant Function

Let $f(t) = a$ (a constant). Then from the definition of the Laplace transform,

$$\mathrm{L}\{a\} = \int_0^\infty a\,e^{-st}\,dt = -\frac{a}{s}e^{-st}\Big|_0^\infty = 0 - \left(-\frac{a}{s}\right) = \boxed{\frac{a}{s}}$$

Matlab:

```
clear all; close all
syms a s t
f=a
F=laplace(f,t,s)
```

Çözüm:

f = a

F =a/s

Laplace transform of the exponential function

where $\alpha > 0$. Then,

$$f(t) = e^{-\alpha t}$$

$$F(s) = \int_0^\infty e^{-\alpha t} e^{-st} dt = \int_0^\infty e^{-(\alpha+s)t} dt$$

$$= \frac{e^{-(s+\alpha)t}}{-(s+\alpha)}\Bigg]_0^\infty = 0 - \frac{e^{-0}}{-(s+\alpha)}$$

$$= \frac{1}{s+\alpha}$$

Matlab:

```
clear all; close all
syms a t
f = exp(-a*t);
F=laplace(f)
```

Çözüm:

```
F = 1/(a + s)
```

Note: Euler's formula

$$e^{j\omega_0 t} = \cos(\omega_0 t) + j\sin(\omega_0 t)$$

$$\frac{1}{s - j\omega_0} \qquad \frac{s}{s^2 + \omega_0^2} \qquad j\frac{\omega_0}{s^2 + \omega_0^2}$$

$$= \frac{1}{s - j\omega_0}\frac{s + j\omega_0}{s + j\omega_0}$$

$$= \frac{s - j\omega_0}{s^2 + \omega_0^2}$$

Some Functions f(t) and Their Laplace Transforms L(f)

$f(t)$	$F(s) = L[f(t)]$
1 or $u(t)$	$\frac{1}{s}$
$e^{-\alpha t}$	$\frac{1}{s+\alpha}$
$\sin \omega t$	$\frac{\omega}{s^2+\omega^2}$
$\cos \omega t$	$\frac{s}{s^2+\omega^2}$
$e^{-\alpha t} \sin \omega t$	$\frac{\omega}{(s+\alpha)^2+\omega^2}$
$e^{-\alpha t} \cos \omega t$	$\frac{s+\alpha}{(s+\alpha)^2+\omega^2}$
t	$\frac{1}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
$e^{-\alpha t} t^n$	$\frac{n!}{(s+\alpha)^{n+1}}$
$\delta(t)$	1

Determine the inverse transform of the function below.

$$F(s) = \frac{5}{s} + \frac{12}{s^2} + \frac{8}{s+3}$$

$$f(t) = 5 + 12t + 8e^{-3t}$$

Matlab:
clear all; close all
syms a s
f =5/s+12/s^2+8/(s+3);
F=ilaplace(f)

Çözüm:
F =12*t + 8*exp(-3*t) + 5

Transforms of Derivatives and Integrals. ODEs

Theorem

Laplace Transform of Integral

Let $F(s)$ denote the transform of a function $f(t)$ which is piecewise continuous for $t \geq 0$, Then, for $s > 0$, $s > k$, and $t > 0$,

$$\mathsf{L}\left\{\int_0^t f(\tau)d\tau\right\} = \frac{1}{s}F(s), \qquad \text{thus} \qquad \int_0^t f(\tau)d\tau = \mathsf{L}^{-1}\left\{\frac{1}{s}F(s)\right\}.$$

Significant Operations for Solving Differential Equations

$$L[f'(t)] = sF(s) - f(0)$$

$$L[f''(t)] = s^2F(s) - sf(0) - f'(0)$$

$$L\left[\int_0^t f(t)dt\right] = \frac{F(s)}{s}$$

f(0) and f'(0), initial condition at $t = 0$

Laplace Transform for ODEs

- Equation with initial conditions

$$\frac{d^2y}{dt^2} + y = 1, \qquad y(0) = y'(0) = 0$$

- Laplace transform is linear

$$\mathcal{L}(y'') + \mathcal{L}(y) = \mathcal{L}(1)$$

- Apply derivative formula

$$s^2\mathcal{L}(y) - sy(0) - y'(0) + \mathcal{L}(y) = \frac{1}{s}$$

- Rearrange

$$\mathcal{L}(y) = \frac{1}{s(s^2+1)} = \frac{1}{s} - \frac{s}{s^2+1}$$

- Take the inverse

$$y = 1 - \cos t$$

Example

$$\frac{d^2y}{dt^2} + 6\frac{dy}{dt} + 8y = 2 \quad y(0) = y'(0) = 0$$

$$s^2 Y(s) + 6sY(s) + 8Y(s) = 2/s$$

$$Y(s) = \frac{2}{s(s+2)(s+4)}$$

$$Y(s) = \frac{1}{4s} + \frac{-1}{2(s+2)} + \frac{1}{4(s+4)}$$

$$y(t) = \frac{1}{4} - \frac{e^{-2t}}{2} + \frac{e^{-4t}}{4}$$

- ODE w/initial conditions
- Apply Laplace transform to each term
- Solve for Y(s)
- Apply partial fraction expansion
- Apply inverse Laplace transform to each term

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