

Algoritma ve Programlama -1

1. HAFTA

Bilgisayar, Yazılım, Programlama Dilleri,
Problem Çözme ve Algoritma Mantığı

Bilgisayar

- Donanım ve yazılımdan oluşan,
- İletişim, etkileşim ve işbirliği amacıyla kullanıcı tarafından girilen verileri alabilen,
- Bu veriler üzerinde aritmetiksel ve mantıksal işlemler yapabilen,
- Daha sonra bu verileri saklayabilen, güncelleyebilen, paylaşabilen,
- Kullanıcı ve işletim sistemi aracılığı ile de yönetebilen veya çıktı olarak sunabilen

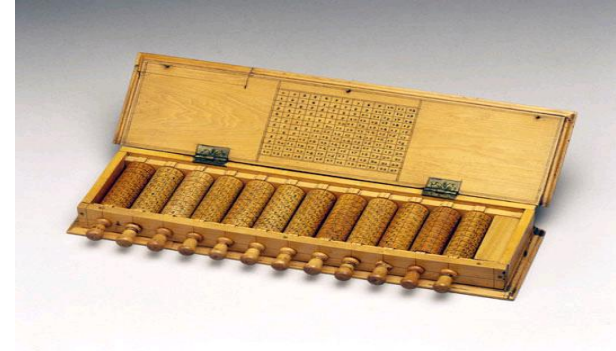
elektronik bir cihazdır [1].

Bilgisayar

- Üç temel özellik [2]:
 - İyi tanımlanmış belirli bir komut setine yanıt verir.
 - Önceden tanımlanmış komut listesini yürütebilir.
 - Çok miktarda veriyi depolayabilir ve okuyabilir.

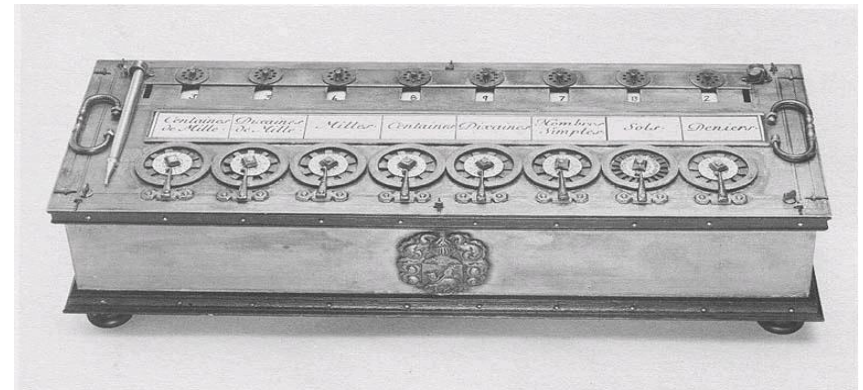
Bilgisayarın Gelişimi

- Napier's Bones , **John Napier**, 1614
 - Çarpma, Bölme, Kare ve Küp Hesaplama



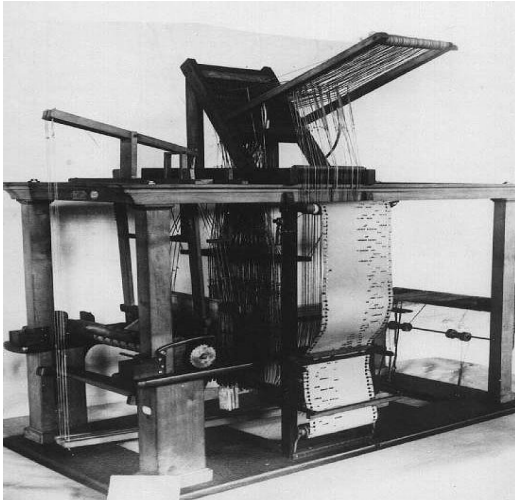
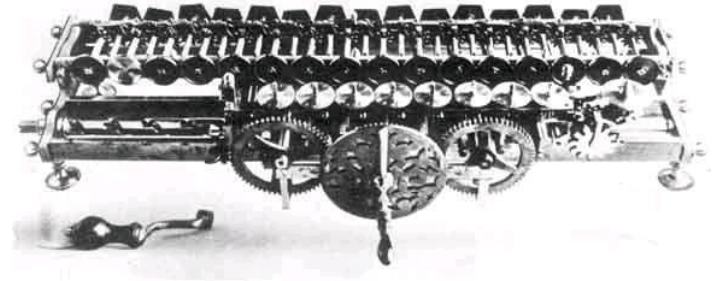
- Slide Rule, **William Oughtred**, 1622
 - Çarpma, Bölme, Kökler, Logaritma, Trigonometri

- Pascaline, **William Pascal**, 1642
 - Toplama, Çıkarma
 - Çarpma , Bölme



Bilgisayarın Gelişimi

- Stepped Reckoner, **G. Wilhelm Leibniz**, 1672
 - Toplama, Çıkarma, Çarpma, Bölme



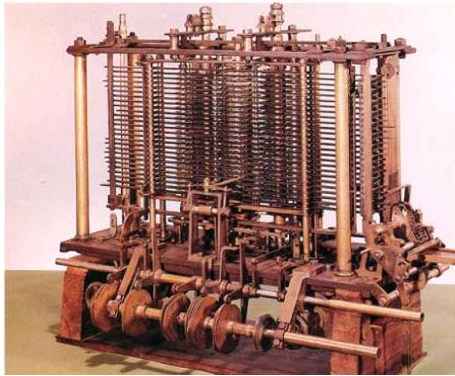
- Jacquard Loom, **Joseph-Marie Jacquard**, 1801
 - Otomatik Tezgah
 - Delikli Kartlar (Punch Cards)



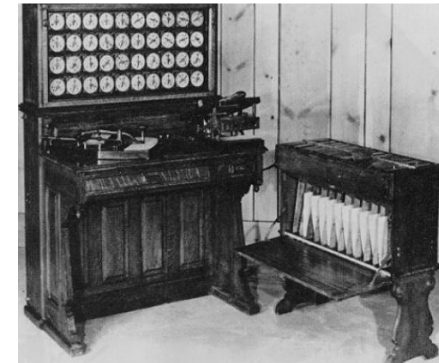
- Bilgisayar tabanlı halı dokuma makinesi

Bilgisayarın Gelişimi

- Difference Engine and Analytical Engine, **Charles Babbage**, 1822 ve 1834
 - İlk mekanik bilgisayar
 - Augusta Ada Byron, binary sistemi önerisi ve analitik motor için ilk program



- Scheutjian Calculation Engine, **Per Georg Scheutz**, 1843
 - Fark Motoru tabanlı
 - İlk baskı hesap makinesi
- Tabulating Machine, **Herman Hollerith**, 1890
 - Delikli kartlarda bulunan bilginin özetlenmesi
 - Hesaplama ve envanter kontrolü
 - Elektromekanik
 - Amerika oy sayımı – 63 milyon , 6 hafta



Bilgisayarın Gelişimi

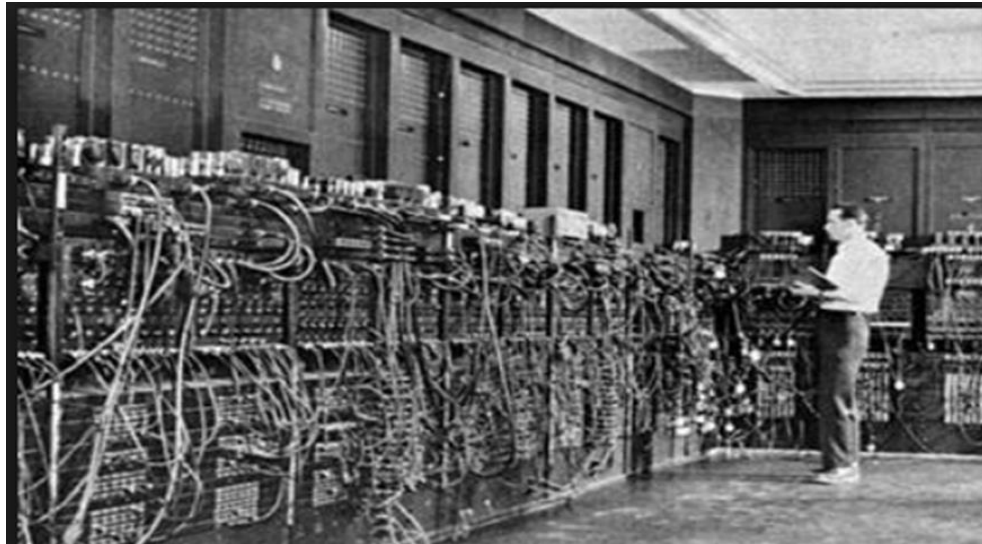
- Atanasoff-Berry Computer , 1939
 - **John Atanasoff** ve lisansüstü öğrencisi **Clifford Berry**,
 - ABC olarak adlandırılan ilk ikili sayı sisteminde çalışan bilgisayar
 - Lojik işlemler için vakum tüpleri
 - Hafıza için kondansatörler



- Boolean Algebra – Bool Cebri , **George Boole**
 - Sadece 1 ve 0 değerlerini bilgisayar devrelerine uygulanması
 - İlk elektronik bilgisayar mantığını ortaya çıkması

Bilgisayarın Gelişimi






- **ENIAC , John Presper Eckert ve John W. Mauchl ,1946**
 - Electronic Numerical Integrator and Computer
 - İlk programlanabilir genel amaçlı elektronik dijital bilgisayar,
 - 167 m2 alan ve 30 ton ağırlık,
 - Toplama, Çıkarma, Çarpma, Bölme, ?



Bilgisayarın Gelişimi

- Birinci Nesil Bilgisayarlar **(1945 - 1956)**
 - Vakum tüpleri,
- İkinci Nesil Bilgisayarlar **(1956 - 1964)**
 - Transistörler,
 - Programlama Dili, Cobol-Fortran
- Üçüncü Nesil Bilgisayarlar **(1964 - 1970)**
 - Entegre Devreler
- Dördüncü Nesil Bilgisayarlar **(1970 - Günümüze)**
 - Integrated Circuit Chip –Bütünleşik Devre Yongası
 - İlk kişisel bilgisayar – PC
 - Yerel ağ bağlantısı – LAN
- Beşinci Nesil Bilgisayarlar **(Günümüzden - Geleceğe)**
 - Paralel işleme – Paralel processing
 - Yapay Zeka
 - Milyarlarca transistör

Computer Generations

Generation	Device	Hardware feature	Characteristics	System names
First (1942-1959)		<ul style="list-style-type: none"> ▶ Vacuum Tubes ▶ Punch Cards 	<ul style="list-style-type: none"> ▶ Support machine language only ▶ Very costly ▶ Generate lot of heat ▶ Huge size ▶ Consumed lot of electricity 	<ul style="list-style-type: none"> ▶ ENIAC ▶ EDVAC ▶ TBM 701
Second (1959-1965)		<ul style="list-style-type: none"> ▶ Transistors ▶ Magnetic Tapes 	<ul style="list-style-type: none"> ▶ Batch operating system ▶ Faster, smaller and reliable than previous generation ▶ Costly 	<ul style="list-style-type: none"> ▶ Honeywell 400 ▶ CDC 1604 ▶ IBM 7030
Third (1965-1975)		<ul style="list-style-type: none"> ▶ ICs ▶ Large capacity disk and Magnetic Tapes 	<ul style="list-style-type: none"> ▶ Time Sharing OS ▶ Faster, smaller and reliable cheaper ▶ Easier to update 	<ul style="list-style-type: none"> ▶ IBM 360/370 ▶ CDC 6600 ▶ PDP 8/11
Fourth (1975-1988)		<ul style="list-style-type: none"> ▶ ICs with VLSI Technology ▶ Semiconductor Memory ▶ Magnetic tapes and floppy as portable 	<ul style="list-style-type: none"> ▶ Multiprocessing & GUI OS ▶ Object oriented programs ▶ Small, affordable, easy to Use ▶ Easier to update 	<ul style="list-style-type: none"> ▶ Apple II ▶ VAX 9000 ▶ CRAY 1/2
Fifth (1988-Present)		<ul style="list-style-type: none"> ▶ ICs with ULSI Technology ▶ Large capacity hard disk with RAID Support ▶ Optical disks as portable read-only storage media ▶ powerful servers, internet, Cluster computing 	<ul style="list-style-type: none"> ▶ Powerful, cheaper, reliable, easy to use, portable ▶ Rapid software development possible 	<ul style="list-style-type: none"> ▶ IBM ▶ Pentium ▶ PARAM

EnableTips.com

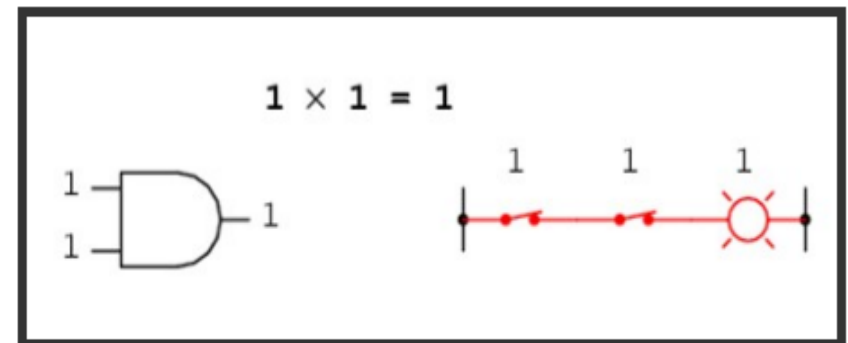
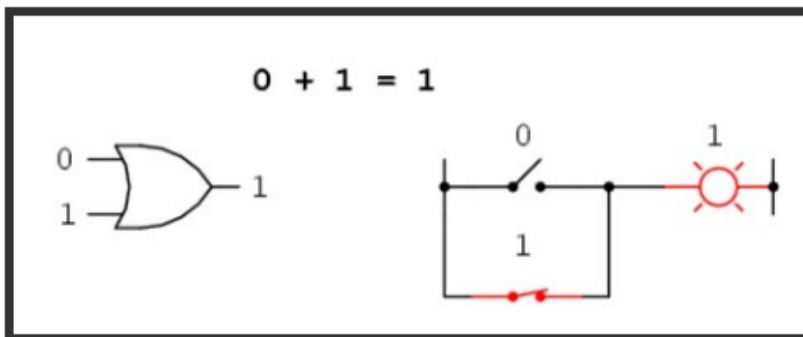
Sayı Sistemleri

- Günlük hayat?
- Bilgisayar hangi sayı sistemini kullanır?
- Neden?

*Not: Sayı sistemleri, 1.Hafta Lab dersinde anlatılacaktır.

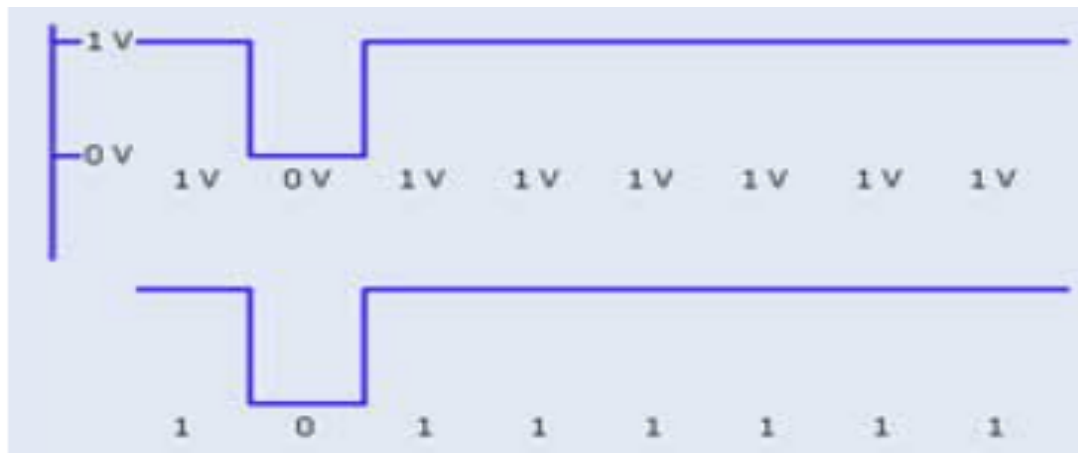
Neden İkili Sayı Sistemi?

- Digital vs Analog
- Eğer elektrik akımı varsa, anlık durum 1 değerini alır, yoksa 0 değerini alır.
- Boole cebiri



Neden İkili Sayı Sistemi

- İkili sayı sistemindeki her bir bit veya rakam tek bir 0 veya 1'dir.
- Bir bit anlaşmak için yeterli mi?
- Neden ondalık sayı sistemi kullanılmıyor?
- Bit - Açılımı?
 - Binary Digit



Karakter Setleri

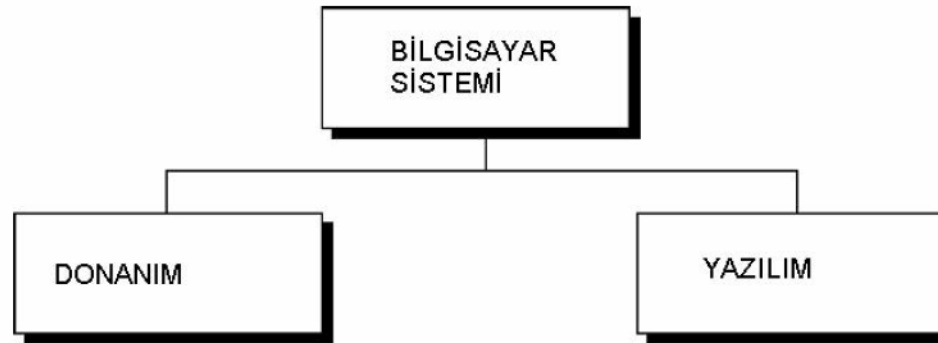
- Bilgisayarlar yalnızca sayılarla çalışırlar.
- Harflere ve diğer semboller?
 - Sayılara karşılık düşürülecek biçimde kodlanırlar.
- **ASCII**(**A**merican **N**ational **C**ode for **I**nformation **I**nterchange) , ANSI (American National Standards Institute), 1963
- ASCII - 7 bit, Kaç karakterden oluşur?
- Extended ASCII – 8 bit
- Unicode
- 1 Byte
 - Kaç bit?
 - Neden?

*Not: Karakter setleri, 1.Hafta Lab. dersinde anlatılacaktır.

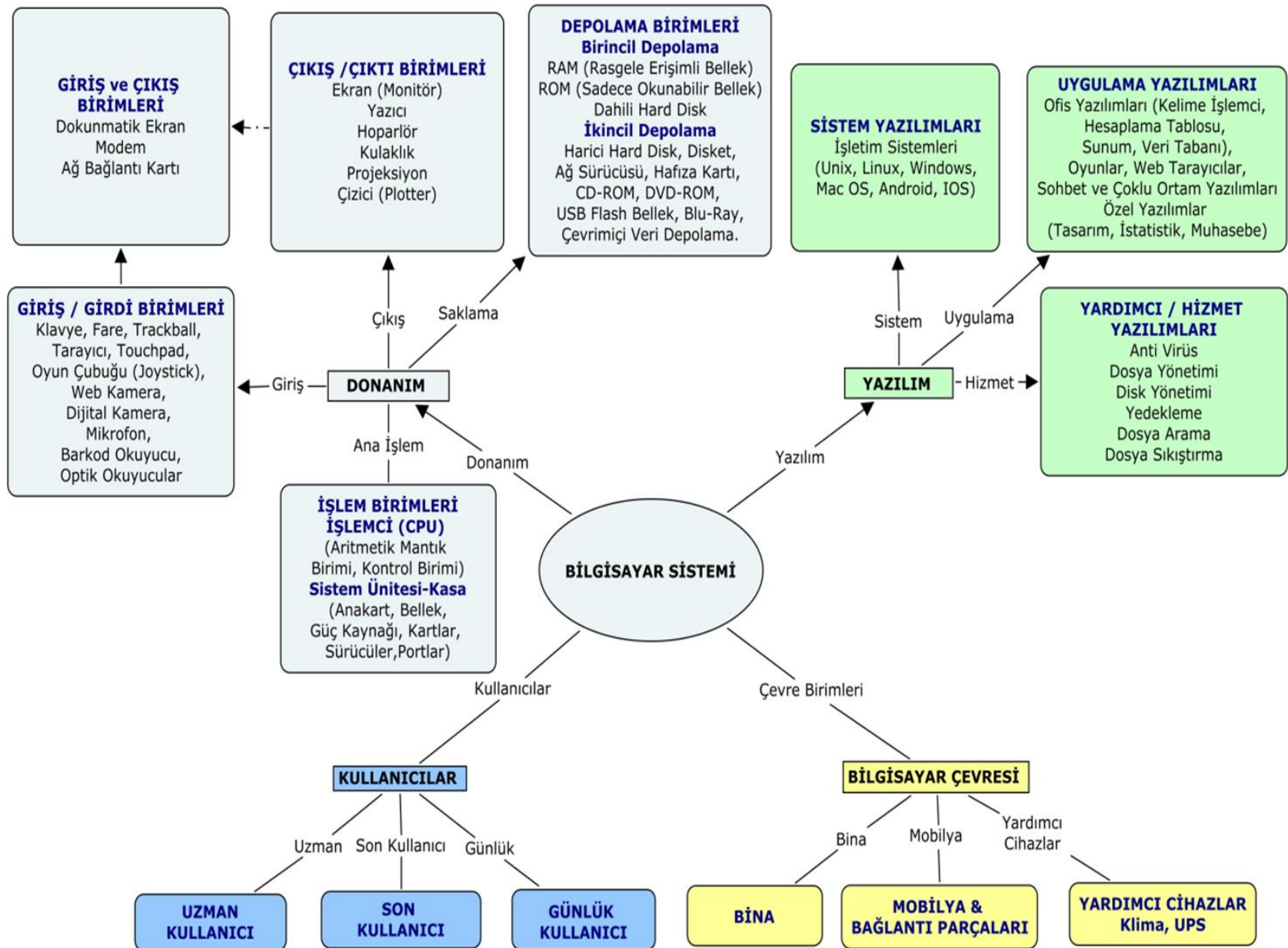
Geniřletilmiř ASCII Karakter Seti

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
1	01	Start of heading	33	21	!	65	41	A	97	61	a	129	81	ü	161	A1	í	193	C1	ł	225	E1	β
2	02	Start of text	34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	Ł	226	E2	Γ
3	03	End of text	35	23	#	67	43	C	99	63	c	131	83	â	163	A3	ú	195	C3	ł	227	E3	π
4	04	End of transmit	36	24	\$	68	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
5	05	Enquiry	37	25	%	69	45	E	101	65	e	133	85	à	165	A5	Ñ	197	C5	†	229	E5	σ
6	06	Acknowledge	38	26	&	70	46	F	102	66	f	134	86	å	166	A6	ª	198	C6	‡	230	E6	μ
7	07	Audible bell	39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	º	199	C7	‡	231	E7	ι
8	08	Backspace	40	28	(72	48	H	104	68	h	136	88	ê	168	A8	¿	200	C8	Ł	232	E8	Φ
9	09	Horizontal tab	41	29)	73	49	I	105	69	i	137	89	ë	169	A9	ƒ	201	C9	ƒ	233	E9	Θ
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	¬	202	CA	Ł	234	EA	Ω
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k	139	8B	ï	171	AB	½	203	CB	ƒ	235	EB	Θ
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l	140	8C	î	172	AC	¾	204	CC	‡	236	EC	∞
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m	141	8D	ì	173	AD	ı	205	CD	=	237	ED	ø
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n	142	8E	Ë	174	AE	«	206	CE	‡	238	EE	ε
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o	143	8F	Ä	175	AF	»	207	CF	Ł	239	EF	Π
16	10	Data link escape	48	30	0	80	50	P	112	70	p	144	90	É	176	B0	☐	208	DO	Ł	240	FO	≡
17	11	Device control 1	49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	☐	209	D1	ƒ	241	F1	±
18	12	Device control 2	50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	☐	210	D2	ƒ	242	F2	≥
19	13	Device control 3	51	33	3	83	53	S	115	73	s	147	93	ô	179	B3		211	D3	Ł	243	F3	≤
20	14	Device control 4	52	34	4	84	54	T	116	74	t	148	94	ö	180	B4		212	D4	Ł	244	F4	[
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u	149	95	ò	181	B5		213	D5	ƒ	245	F5]
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v	150	96	û	182	B6		214	D6	ƒ	246	F6	÷
23	17	End trans. block	55	37	7	87	57	W	119	77	w	151	97	ù	183	B7		215	D7	‡	247	F7	≈
24	18	Cancel	56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8		216	D8	‡	248	F8	°
25	19	End of medium	57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9		217	D9	ƒ	249	F9	•
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	186	BA		218	DA	ƒ	250	FA	·
27	1B	Escape	59	3B	;	91	5B	[123	7B	{	155	9B	÷	187	BB		219	DB	■	251	FB	√
28	1C	File separator	60	3C	<	92	5C	\	124	7C		156	9C	£	188	BC		220	DC	■	252	FC	¤
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}	157	9D	¥	189	BD		221	DD	■	253	FD	z
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~	158	9E	ℳ	190	BE		222	DE	■	254	FE	■
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□	159	9F	f	191	BF		223	DF	■	255	FF	□

Bilgisayar Sistemi

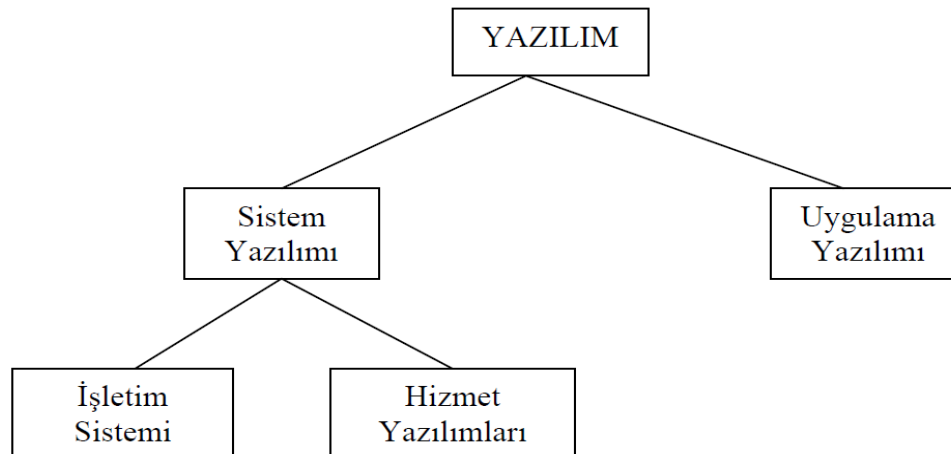


- Donanım
 - Her türlü fiziksel parça
- Yazılım
 - Kullanıcıya ait her türlü işlemi gerçekleştirmek için donanıma komutlar veren programlar
- Bilgisayar Sistemi
 - İşlem, giriş, çıkış ve depolama birimleri
 - Kullanıcılar
 - Çevre
 - Tüm bunları birbirine bağlayan aygıtlar



Yazılım

- Bilgisayar yazılımı
 - Sistem yazılımları (System Software)
 - Uygulama yazılımları (Application Software)
- Sistem yazılımları
 - İşletim Sistemleri – Windows, Linux, OS X , Pardus vb.
 - Hizmet Yazılımları – Disk birleştirme, Disk Temizleme, Veri Sıkıştırma, Yedekleme vb.
- Uygulama yazılımları
 - Genel Amaçlı – Kelime İşlem, Hesap Tablosu, Sunum, Veritabanı Yönetim
 - Özel Amaçlı – Analiz ve Karar verme, Ticari yazılımlar, Eğlence , Eğitim



İşletim Sistemleri

- Bir bilgisayarda işlemlerin yapılabilmesini,
- Komutların yerine getirilmesini,
- Programların birbiriyle uyum içerisinde sorunsuz çalışmasını,
- Kullanacakları disk alanını ve araçların paylaşımını düzenleyen,

kısaca bilgisayarın temel işlevini yerine getirebilmesini sağlayan yazılımlardır.



İşletim Sistemleri



- Bir yazı yazdınız ve diske kaydediyorsunuz.
- Kelime işlemci, doğrudan diskle ilgili bir iş yapmaz, sadece işletim sisteminin diskle ilgili fonksiyonlarını kullanır.

Tanım

- Algoritma - Algorithm
- Program
- Programlama - Programming
- Yazılım - Software
- Uygulama – Application
- Programlama Dilleri – Programming Language

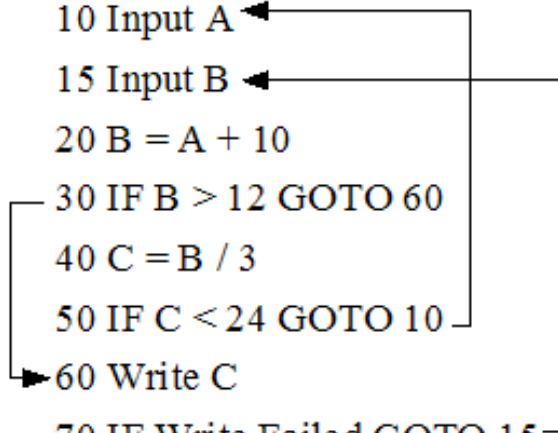
Yapısal ve Yapısal Olmayan Diller

```
IF (A < B) THEN
    C = A
ELSE
    C = B
END IF

FOR I = 1 to 10 DO
    Array[I] = C * 2
END FOR

WHILE (Not End Of File)
    Read line from file
END WHILE
```

```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```

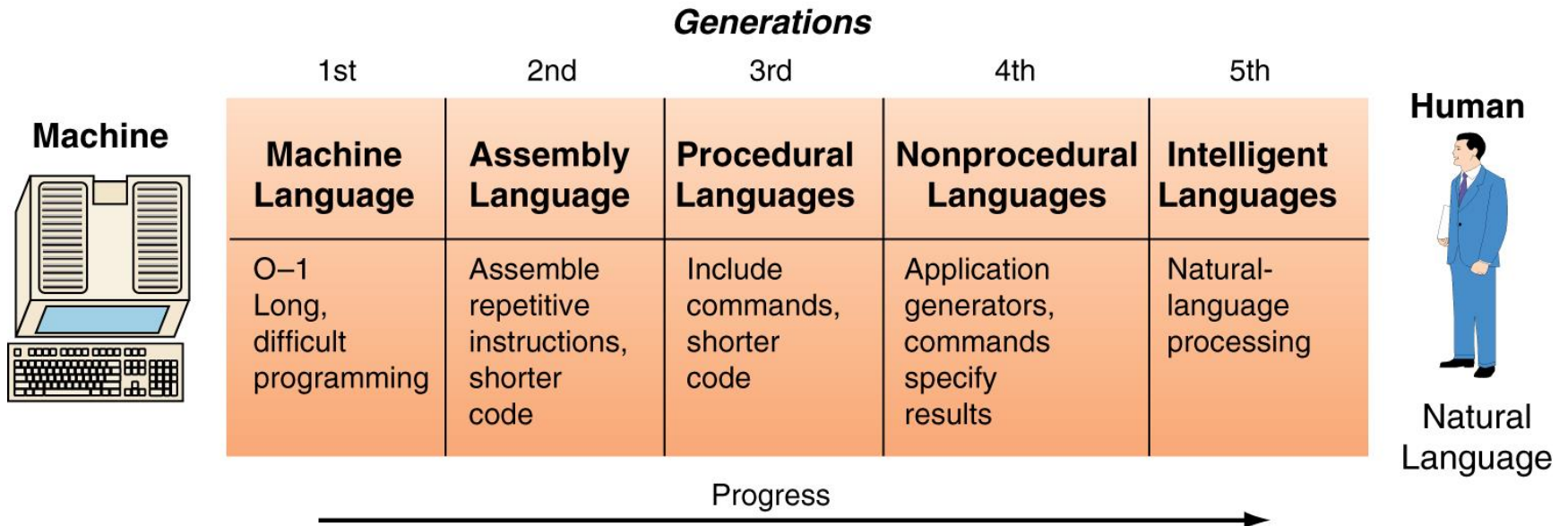


- C, JAVA, PYTHON, C# vb.
- ASSEMBLY, BASIC, COBOL, FORTRAN

Structured and Non-Structured Programming

Structured vs Unstructured Programming	
Structured Programming is a programming paradigm which divides the code into modules or function.	Unstructured Programming is the paradigm in which the code is considered as one single block.
Readability	
Structured Programming based programs are easy to read.	Unstructured Programming based programs are hard to read.
Purpose	
Structured Programming is to make the code more efficient and easier to understand.	Unstructured programming is just to program to solve the problem. It does not create a logical structure.
Complexity	
Structured Programming is easier because of modules.	Unstructured programming is harder when comparing with the structured programming.
Application	
Structured programming can be used for small and medium scale projects.	Unstructured programming is not applicable for medium and complex projects.
Modification	
It is easy to do changes in Structured Programming.	It is hard to do modifications in Unstructured Programming.
Data Types	
Structured programming uses many data types.	Unstructured programming has a limited number of data types.
Code Duplication	
Structured programming avoids code duplication.	Unstructured programming can have code duplication.
Testing and Debug	
It is easy to do testing and debugging in Structured Programming.	It is hard to do testing and debugging in Unstructured programming.

Programlama Dilleri



Machine Language

```
00100101 11010011
00100100 11010100
10001010 01001001 11110000
01000100 01010100
01001000 10100111 10100011
11100101 10101011 00000010
00101001
11010101
11010100 10101000
10010001 01000100
```

Assembly Language

```
ST 1,[801]
ST 0,[802]
TOP: BEQ [802],10,BOT
      INCR [802]
      MUL [801],2,[803]
      ST [803],[801]
      JMP TOP
BOT: LD A,[801]
      CALL PRINT
```

C Language

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num,i,sum=0;
    printf("Enter a number\n");
    scanf("%d",&num);
    for(i=0;i<=num;i++)
    {
        if((i%2)==0)
            continue;
        else
            sum=sum+i;
    }
    printf("Sum of odd number %d",sum);
    getch();
}
```

Procedural and Non-Procedural Languages

Difference between procedural and non-procedural language:

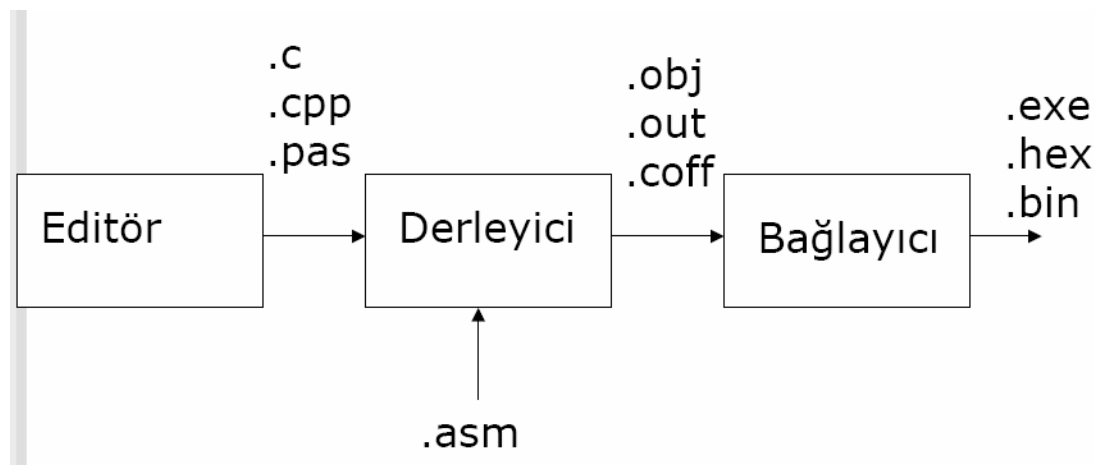
Procedural language	Non-procedural language
Procedural language is a traditional programming language in logical step-by-step process for solving a problem is to specified.	In non-procedural programming language, programmers and users specify the results they require, but do not specify how to do.
Procedural language are based on algorithms	Non procedural language are not based on algorithms
In procedural language, the output as well as the procedures or steps followed is important.	In non-procedural language, the output of the programming is only the focus area.
Procedural language is very lengthy and time taking but it is a very efficient language.	Non-procedural language is short in writing and replaces many lines into single one.
Example of procedural languages are Assembler, Fortran, Cobol, C, etc.	Example of non-procedural languages are SQL, Visual Basic, etc.

Yazılım Geliştirme Araçları

- Editörler
 - Metin düzenleyici
 - Hatalar, anahtar ifadeler belirli değil
 - Notepad, Wordpad
- Tümleşik Geliştirme Ortamları – IDE
 - Derleyici, bağlayıcı
 - Hata ayıklama
 - Gözlem penceresi
 - CodeBlocks, NetBeans, Eclipse vb.

Yazılım Geliştirme Araçları

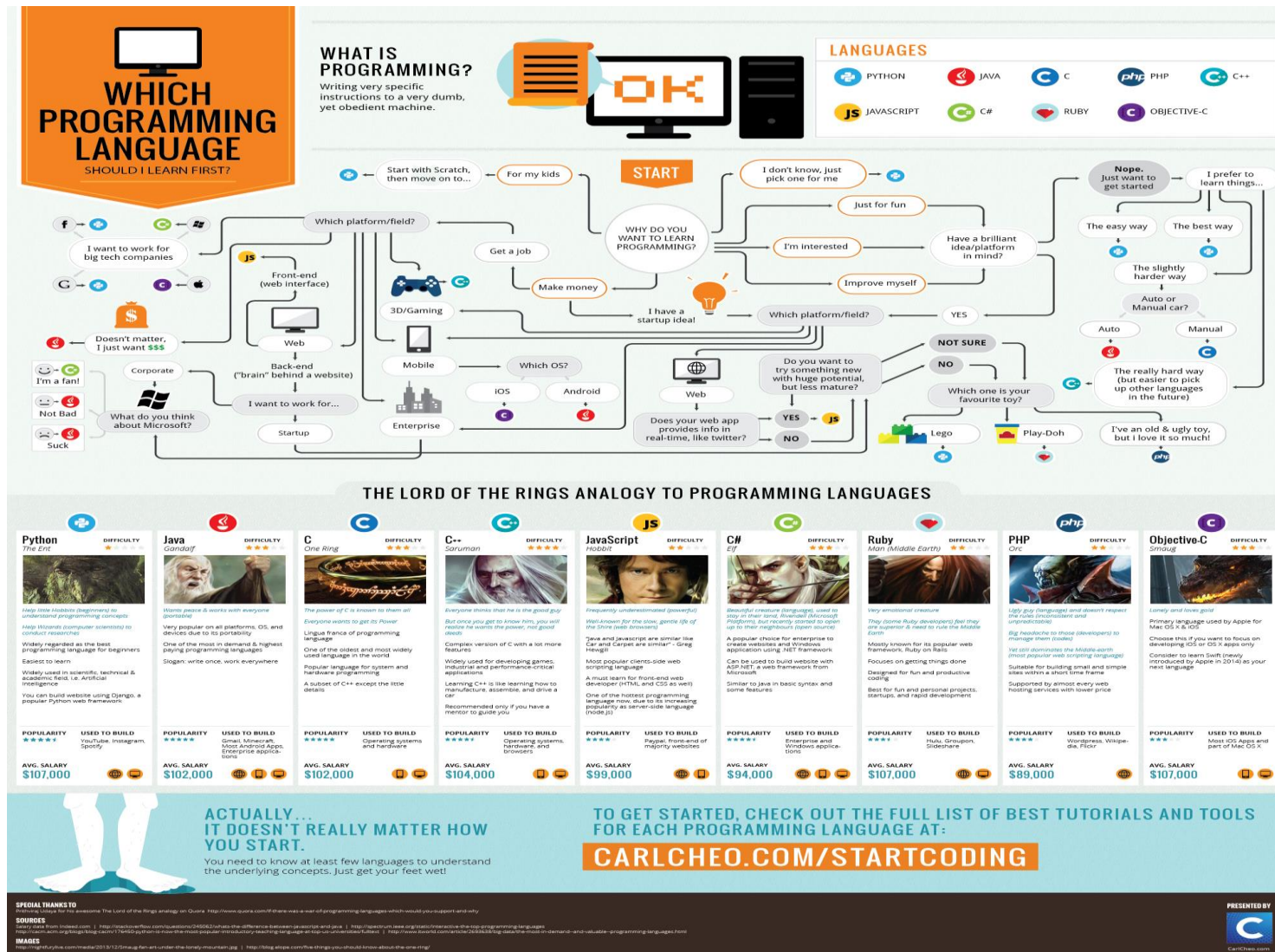
- Derleyiciler
 - Program kodları → Makinenin anlayabileceği OBJ kodlara
- Bağlayıcılar
 - OBJ kodlar → EXE - çalıştırabilir program
- Yorumlayıcılar
 - Program kodunu bir bütün olarak değerlendirmez.
 - Satır, satır yorumlayarak çalıştırırlar.
 - Standart bir amaç kod yok.
 - Derleyicilere göre daha kolay yazılır
 - İcra zamanı derleyiciye göre daha uzun



Scripting Languages and Programming Languages

- Basically, all scripting languages are programming languages.
- The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted.
- For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.
- Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.
- Some programming languages traditionally used with an explicit compilation step are C, C++.

WHICH PROGRAMMING LANGUAGE ?



Problem Çözme - Descartes

- Problem Çözme Tekniği (Descartes'e göre):
 1. Doğruluğu kesin olarak kanıtlanmadıkça, hiçbir şeyi doğru olarak kabul etmeyin; tahmin ve önyargılardan kaçının.
 2. Karşılaştığınız her güçlüğü mümkün olduğu kadar çok parçaya bölün.
 3. Düzenli bir biçimde düşünün; anlaşılması en kolay olan şeylerle başlayıp yavaş yavaş daha zor ve karmaşık olanlara doğru ilerleyin.
 4. Olaya bakışınız çok genel, hazırladığınız ayrıntılı liste ise hiçbir şeyi dışarıda bırakmayacak kadar kusursuz ve eksiksiz olsun.

Problem Çözme - George Polya

- How To Solve It , 1945 George Polya, most prized publication
- One million copies and translated into 17 languages
- Identity four basic principles of problem solving
- Polya's Principles – How to solve it ?
 1. Understand the problem – Problemi anlama
 2. Devise a plan – Planı tasarlama
 3. Carry out the plan – Planı gerçekleştirme
 4. Look back- Geriye bak

Problemi Anlama

- Problemi kısmen ya da tam anlamamak → Gereksiz uğraş
- Polya, öğrencilerin sorması gereken şu gibi sorular belirledi.
 - Sorunu açıklarken kullanılan tüm kelimeleri anlıyor musunuz?
 - Ne bulmayı ya da göstermeyi istiyorsun?
 - Sorunu kendi kelimelerinizle yeniden ifade edebilir misiniz?
 - Sorunu anlamınıza yardımcı olabilecek bir resim veya şema düşünebiliyor musunuz?
 - Bir çözüm bulmanızı sağlayacak yeterli bilgi var mı?

Planı Tasarlama

- Problemleri çözmek için birçok makul yol var.
- Uygun bir strateji seçme becerisi en çok problemi çözerek öğrenilir. Bir stratejiyi seçmek kolaylaşacaktır.
 - Tahmin ve kontrol
 - Örnek ara
 - Düzenli bir liste yap
 - Bir resim çiz
 - Olasılıklar elemine et
 - Daha basit bir problemi çöz
 - Simetri kullan
 - Bir model belirle
 - Özel durumları düşün
 - Geriye doğru çalış
 - Doğrudan akıl yürüt
 - Bir formül kullan
 - Denklem ile çöz
 - Usta ol

Planı Gerçekleştirme

- Genellikle planı tasarlamaktan daha kolay
- İhtiyaç olan becerilere sahip olma, dikkat ve sabır
- Seçilen plan ile devam et
- Çalışmamaya devam ederse onu at ve başka birini seç

Geriye Bakmak

- Ne yaptığınızı, neyin işe yaradığını ve neyi yapmadığınızı yansıtın ve geriye bakın. Bu daha çok zaman kazandıracaktır.
- Bunu yapmak, gelecekteki sorunları çözmek için hangi stratejiyi kullanacağınızı tahmin etmenizi sağlayacaktır.

Problem-1

Kurt, Kuzu, Ot

- Ahmet Amca istiyor ki;
 - Kurt, Kuzu ve Ot derenin karşısına geçecek,
 - Kayık her seferinde birini taşıyacak,
 - Kurt-Kuzu ve Kuzu-Ot baş başa kalmayacak,



Karşıya nasıl geçerler?

Problem-2 / Sayı Bulma

- Sayı hakkında ipuçları;
 - 6 basamaklıdır.
 - 5' e bölünür ama 8'e bölünemez.
 - Onbinler basamağı ile binler basamağı toplamı = 3
 - Onbinler basamağı ile yüzler basamağı toplamı= 11
 - Binler basamağı ile onlar basamağı toplamı = 1
 - Onlar basamağı ile birler basamağı toplamı = 5
 - Yüzbinler basamağı ile onbinler basamağı birbirine eşittir.
 - Sayı 300,000'den küçüktür.

Bu sayı kaçtır?

Problem-3

- Simon diyor ki;
 - 3 lt ve 5 lt 'lik iki kabınız var
 - Kaplarda herhangi ölçüm işareti yok
 - 4 lt' lik suyu nasıl elde edersiniz?



Zor Ölüm 3 – Die Hard with a Vengeance

Problem-3 Çözüm

- Çözüm 1
 - 5lt'lik kabı çeşmeden doldur
 - 5lt'lik kabı 3lt'lik kaba aktar, 5lt'lik kaptan 2lt kalır
 - 3lt'lik kabı boşalt
 - 5lt'lik kaptaki 2lt 'yi 3lt lik kaba aktar, 3lt'lik kaptan 1lt olur
 - 5lt'lik kabı çeşmeden doldur
 - 3lt'lik kaptaki 1lt' yi 5lt 'lik kaba aktar
 - 3lt'lik kabı çeşmeden doldur
 - 5lt'lik kaptaki 2lt üzerine , 3lt'lik kaptaki suyu ilave et
 - 5lt'lik kaptan 4lt olur
- Çözüm 2
 - 3lt'lik kabı çeşmeden doldur
 - 3lt'yi 5lt 'lik kaba aktar
 - 3lt'lik kabı çeşmeden doldur
 - 3lt'yi 5lt'ye doldur, 3lt' lik kaptan 1lt kalır
 - 5lt'lik kabı boşalt
 - 3lt'lik kaptaki 1lt' yi 5lt 'lik kaba aktar
 - 3lt'lik kabı çeşmeden doldur
 - 5lt'lik kaptaki 2lt üzerine , 3lt'lik kaptaki suyu ilave et
 - 5lt'lik kaptan 4lt olur

Problem-3 – Generalised Form

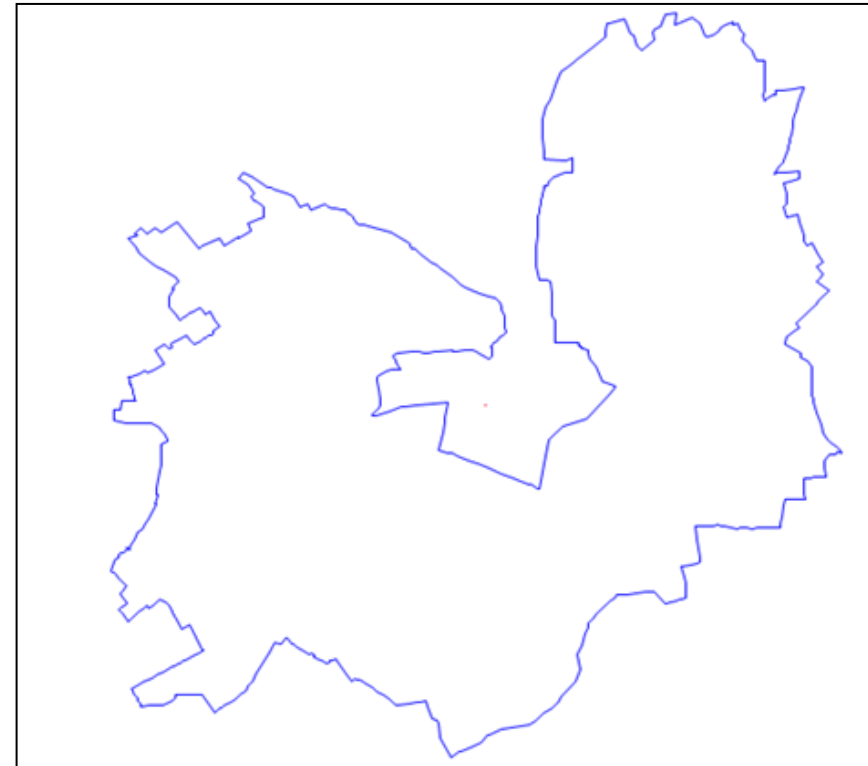
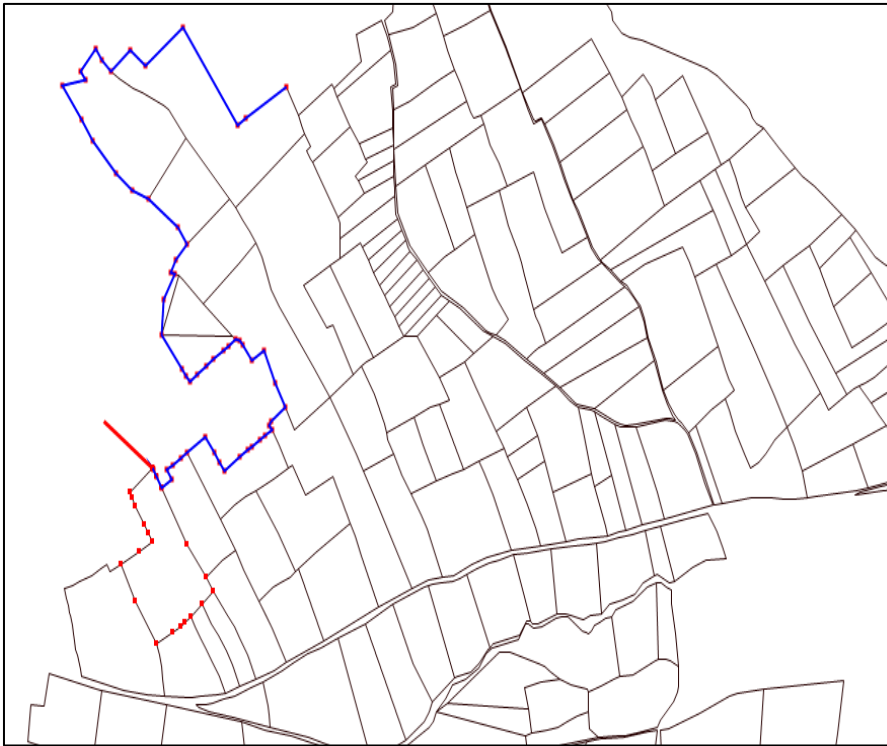
- We have two solutions above which both give the same answer of 3 litres.
- It turns out that if you make it algebraic the answer does not have the same form and it's sort of coincidence.
- Lets call the smaller container A and the larger B and work it through.
- Number 1:
 - Empty the 5 litre can into the 3 litre can - leaving 2 litres in the 5 litre can. ie $B-A$ in B
 - Fill the 3 litre can with the 2 litres from the 5 litre can - leaving 2 litres in the 3 litre can. As in space in A of $A - (B - A) = 2A - B$
 - Fill the remaining 1 litre space in the 3 litre can from the 5 litre can. Leaving 4 litres in the 5 litre can. So the amount in B is $B - (2A - B)$ or $2B - 2A$
- So the generalised form of solution 1 is $2B - 2A$ in B.

Problem-3 – Generalised Form

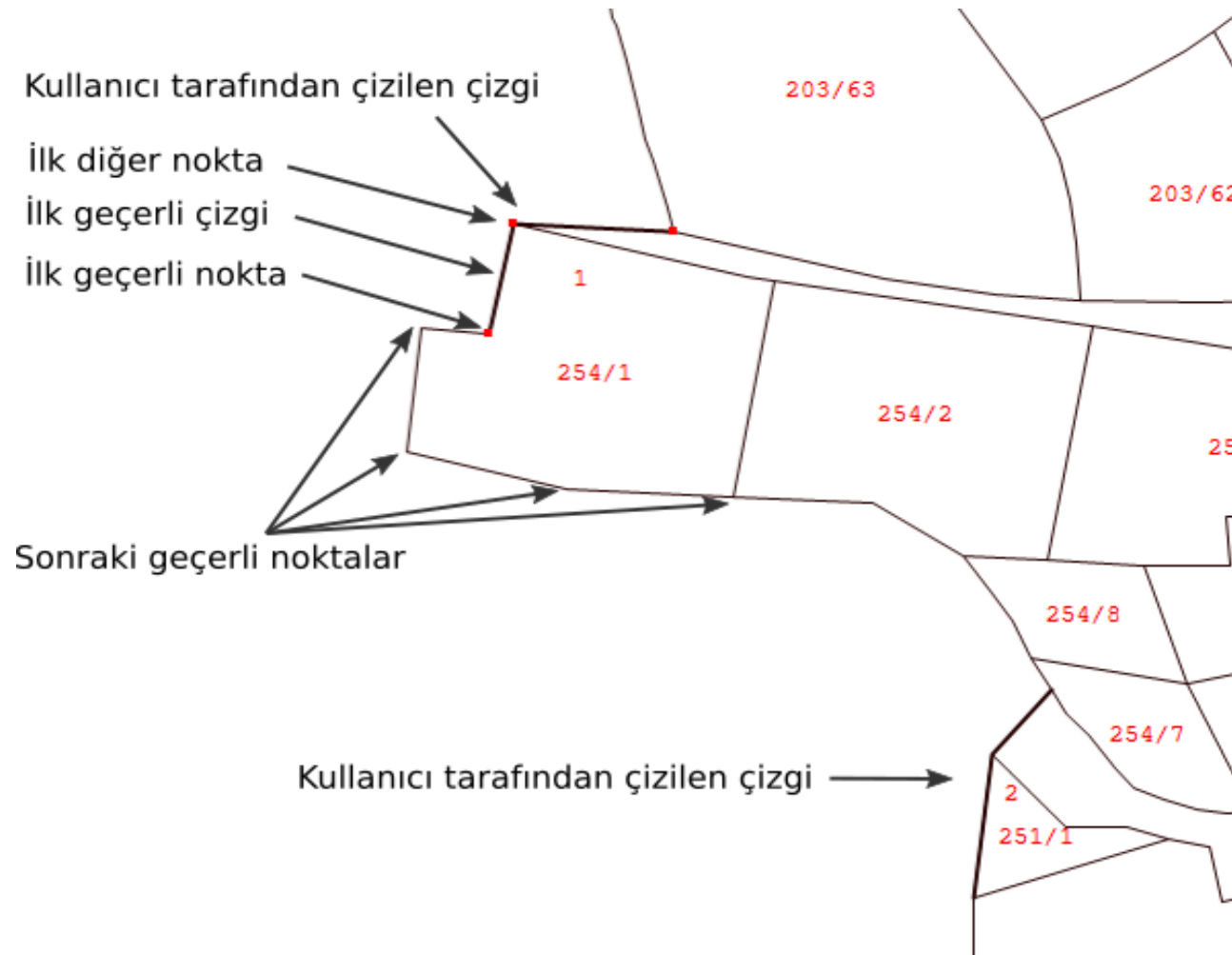
- Number 2:
 - Fill the 3 litre can from the tap. Empty the contents of the 3 litre can into the 5 litre can. Which gives us $B - A$ space in B.
 - Fill the 3 litre can from the tap. Empty the contents of the 3 litre can into the 5 litre can. - Leaving the 5 litre can full and 1 litre in the 3 litre can. ie. removing $(B - A)$ from A leaves $A - (B - A)$ or $2A - B$ in A
 - Pour away the contents of the 5 litre can. Pour the 1 litre from the 3 litre can into the 5 litre can. ie $2A - B$ in
 - Fill the 3 litre can from the tap. Empty the contents of the 3 litre can into the 5 litre can. Leaving 4 litres in the 5 litre can. ie $A + (2A - B)$ or $3A - B$ in B.
- So the generalised form of solution 2 is $3A - B$ in B.

Problem-4

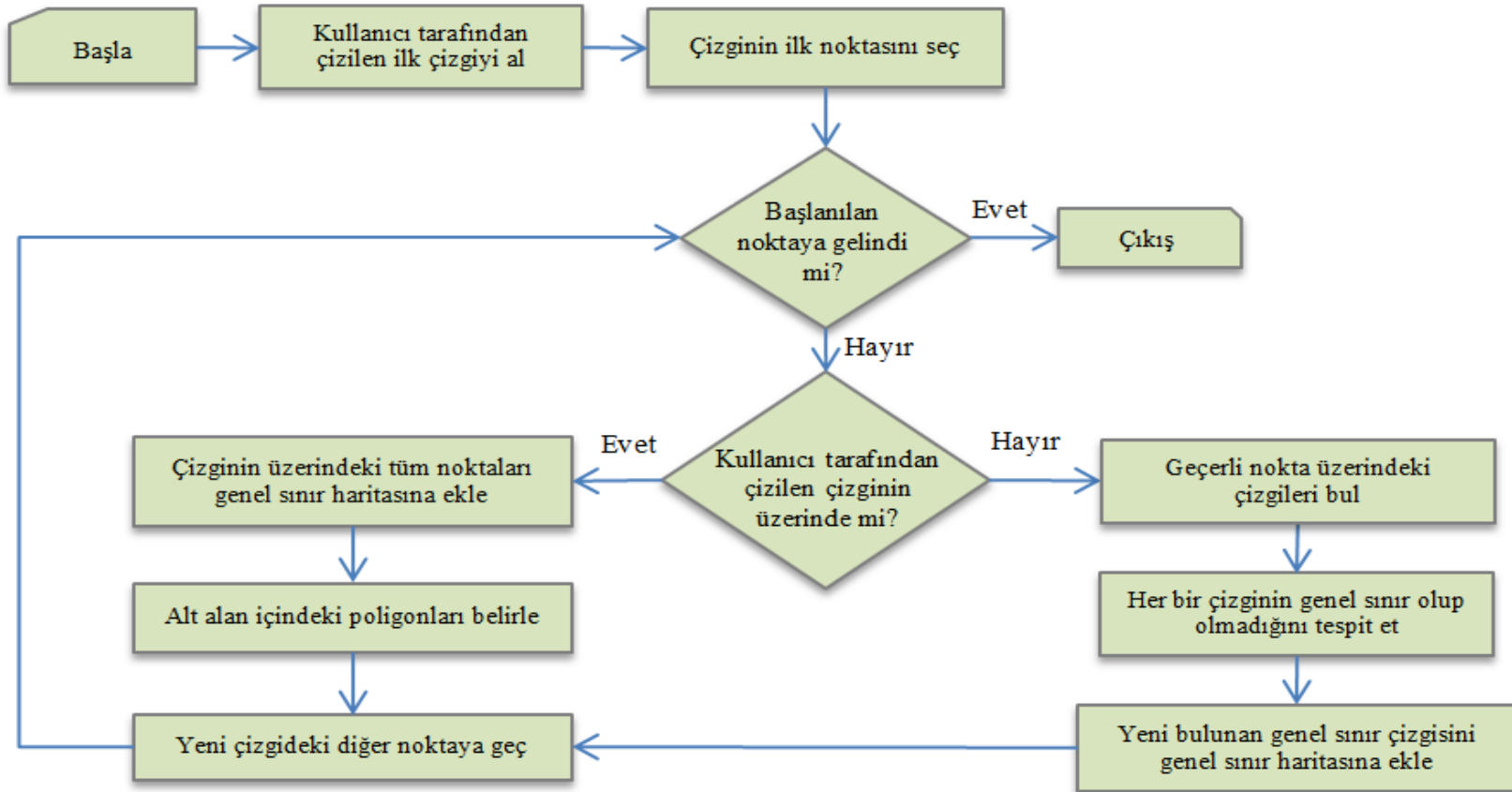
- Genel Sınır Problemi
 - Her kırık nokta tespit edilecek
 - Yol ve kanal ayrımları birleştirilecek
 - Sadece dış kenarlar seçilecek



Problem-4 Çözüm



Problem-4 Çözüm

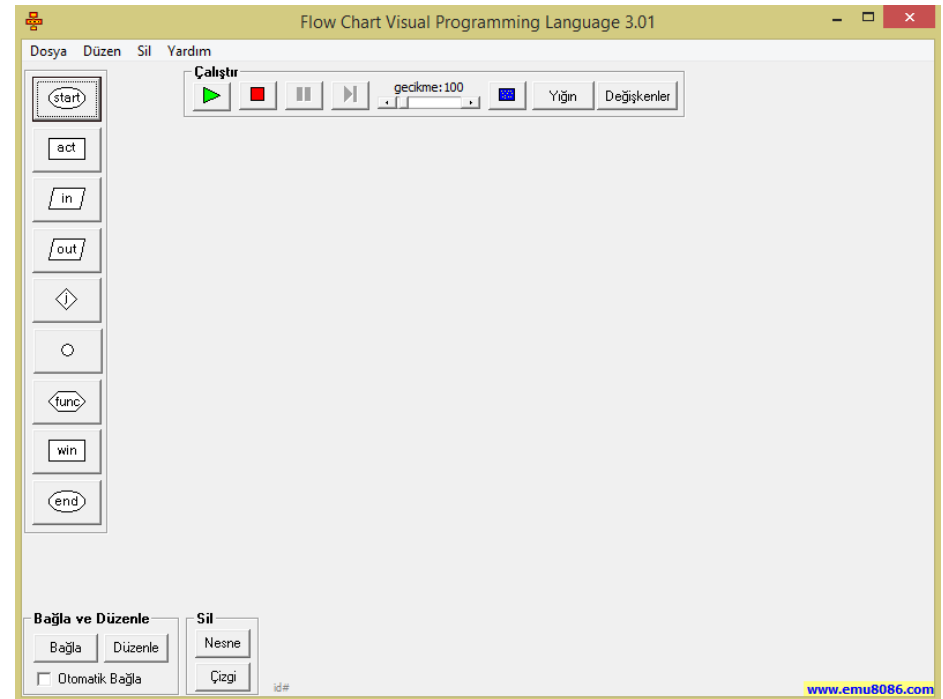


Problem Çözme - Yazılım

- Problem çözme sırası
 - 1. Problemi anlama (Understanding, Analyzing),**
 - 2. Bir çözüm yolu geliştirme (Designing),**
 - 3. Algoritma ve program yazma (Writing),**
 - 4. Tekrar tekrar test etme (Reviewing)**

Önümüzdeki Hafta

- Yazılım Geliştirme Aşamaları
- Algoritma Temsilleri
 - Formüller
 - Kelimeler
 - Sözde Kod
 - Akış Diyagramı
- Algoritma kurma
- Akış Diyagramı Örnekler
 - Flow Chart Visual Programming Language



Kaynakça

1. Başar, E., Bilgi ve İletişim Teknolojileri 1 Ders Notu, MEB EğiTek, Açık Öğretim Okulları, Ankara, 2009.
2. http://www.ftms.edu.my/images/Document/CSCA0101%20-%20Computing%20Basics/csc0101_ch01.pdf
3. <https://www.enabletips.com/2014/03/computer-generations/>
4. http://www.innovit.org/4AI/sicurezza_reti.htm
5. Matunga, O., Micro Computer Architecture, B12 33383 Bsc. Hons. Comp Science.
6. Ayten, U. E., Algoritma ve Programlama Ders Notları, Yıldız Teknik Üniversitesi, 2010.
7. Ataç, S. ,TBT Ders Notu, Dokuz Eylül Üniversitesi Bergama Meslek Yüksekokulu, İzmir, 2016
8. <https://www.differencebetween.com/difference-between-structured-and-vs-unstructured-programming/>
9. <http://bansalwiki.blogspot.com/2013/01/4th-generation-language.html>
10. <https://www.slideserve.com/barny/assembly-machine-language>
11. <https://www.chegg.com/homework-help/database-systems-5th-edition-chapter-5-problem-1rq-solution-9780321523068>
12. <https://www.geeksforgeeks.org/whats-the-difference-between-scripting-and-programming-languages/>
13. <http://xpertlab.com/which-programming-language-should-i-learn-first-xpertlab/>
14. <https://ozgurseremet.com/kurt-kuzu-ot-problemi-calisma-kagidi/>
15. <http://puzzles.nigelcoldwell.co.uk/twentytwo.htm>
16. Haklı, H. , 7170197 Nolu Tübitak-1507 Projesi.
17. <https://math.berkeley.edu/~gmelvin/polya.pdf>

Görsel (Resim) Kaynakça

1. <https://mondaynote.com/the-operating-system-fountain-of-youth-ios-39bc1a3ce004>
2. https://www.webopedia.com/TERM/O/operating_system.html
3. http://www.yitsplace.com/Programming/SW_paradigms.htm
4. http://diehard.wikia.com/wiki/Tompkins_Square_Park

Understand the Problem

- First. You have to understand the problem.
- What is the unknown? What are the data? What is the condition?
- Is it possible to satisfy the condition? Is the condition sufficient to determine the unknown? Or is it insufficient? Or redundant? Or contradictory?
- Draw a figure. Introduce suitable notation.
- Separate the various parts of the condition. Can you write them down?

Devising a Plan

- Second. Find the connection between the data and the unknown. You may be obliged to consider auxiliary problems if an immediate connection cannot be found. You should obtain eventually a plan of the solution.
- Have you seen it before? Or have you seen the same problem in a slightly different form?
- Do you know a related problem? Do you know a theorem that could be useful?
- Look at the unknown! Try to think of a familiar problem having the same or a similar unknown.
- Here is a problem related to yours and solved before. Could you use it? Could you use its result? Could you use its method? Should you introduce some auxiliary element in order to make its use possible?
- Could you restate the problem? Could you restate it still differently? Go back to definitions.

Devising a Plan

- If you cannot solve the proposed problem, try to solve first some related problem.
- Could you imagine a more accessible related problem? A more general problem? A more special problem? An analogous problem?
- Could you solve a part of the problem? Keep only a part of the condition, drop the other part; how far is the unknown then determined, how can it vary?
- Could you derive something useful from the data? Could you think of other data appropriate to determine the unknown?
- Could you change the unknown or data, or both if necessary, so that the new unknown and the new data are nearer to each other?
- Did you use all the data? Did you use the whole condition? Have you taken into account all essential notions involved in the problem?

CARRYING OUT THE PLAN

- Third. Carry out your plan.
- Carrying out your plan of the solution, check each step.
- Can you see clearly that the step is correct?
- Can you prove that it is correct?

LOOKING BACK

- Fourth. Examine the solution obtained.
- Can you check the result? Can you check the argument?
- Can you derive the solution differently? Can you see it at a glance?
- Can you use the result, or the method, for some other problem?