

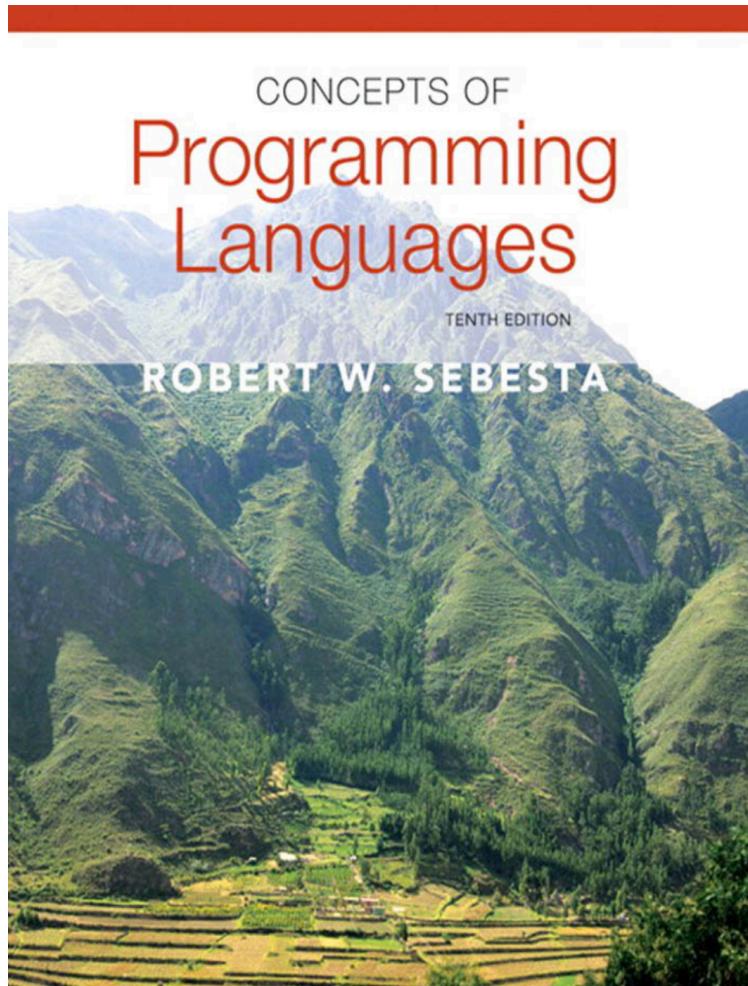
# 315 programlama dilleri

Dr. Ahmet Arif AYDIN

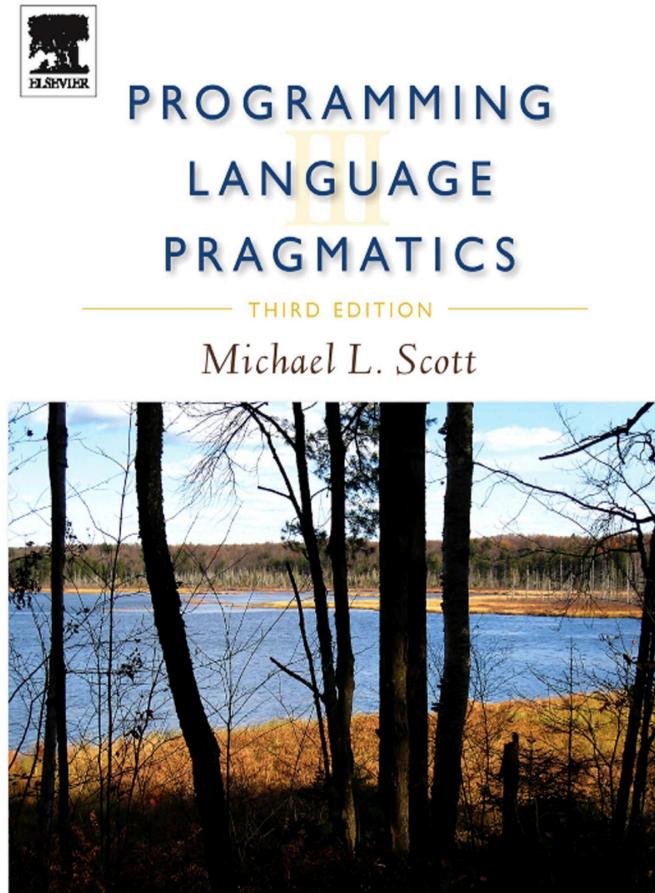
# Hedeflerimiz

- Programlama dillerinin konseptine genel bir bakış
- Kod yazarken kendimizi daha iyi ifade etmek
- Yapacağınız işleme uygun dili seçmede kolaylık
- Yeni bir dil öğrenme becerisi kazandırmak
- Bildiğimiz programlama dillerini daha iyi kullanabilme

# Ders Materyali



<https://cs444pnu1.files.wordpress.com/2014/02/concepts-of-programming-languages-10th-sebesta.pdf>



[https://www.researchgate.net/publication/220695725\\_Programming\\_language\\_pragmatics\\_2\\_ed](https://www.researchgate.net/publication/220695725_Programming_language_pragmatics_2_ed)



# Konular

Hafta	Konular
1	Programlama dillerinin gelişimi
2	Syntax ve semantics kavramları
3	Bağlanmama (Bindings)
4	Tip kontrolü ve alan kontrolü type (checking and scopes)
5	Data tipleri
6	İfade ve atama durumları
7	Durum seviyeli kontrol yapıları
8	Altprogramlar ve altprogram uygulamaları
9	Soyut data tipleri
10	Nesne tabanlı programlama için destekler
11	Fonksiyonel programlama dilleri ve LISP
12	Mantıksal programlama dilleri ve PROLOG
13	Concurrency

# Tavsiyeler

- Dersi ciddiye alın !
- Zamanınızı dikkatli kullanın!
- İletişim kopukluğu olmasın
  - Ofis saatlerini mutlaka değerlendirin!
    - F Blok 2.Kat No:1
    - Perşembe (13:30-17:00)
    - İÖ: Çarşamba (13:30-17:00)
  - Email'i ciddi olarak kullanın!
  - Duyurular, ödevler ve ders notları için
- Sorumluluklarınızı tekrar gözden geçirelim!

[arif.aydin@inonu.edu.tr](mailto:arif.aydin@inonu.edu.tr)

<https://github.com/aaaydin/315PL>

# Puanlama

- Vize
  - Ödevler : 50 p
  - Vize : 50p
- Final
  - Ödevler: 50 p
  - Final 50p



# İlk Bilgisayarlar

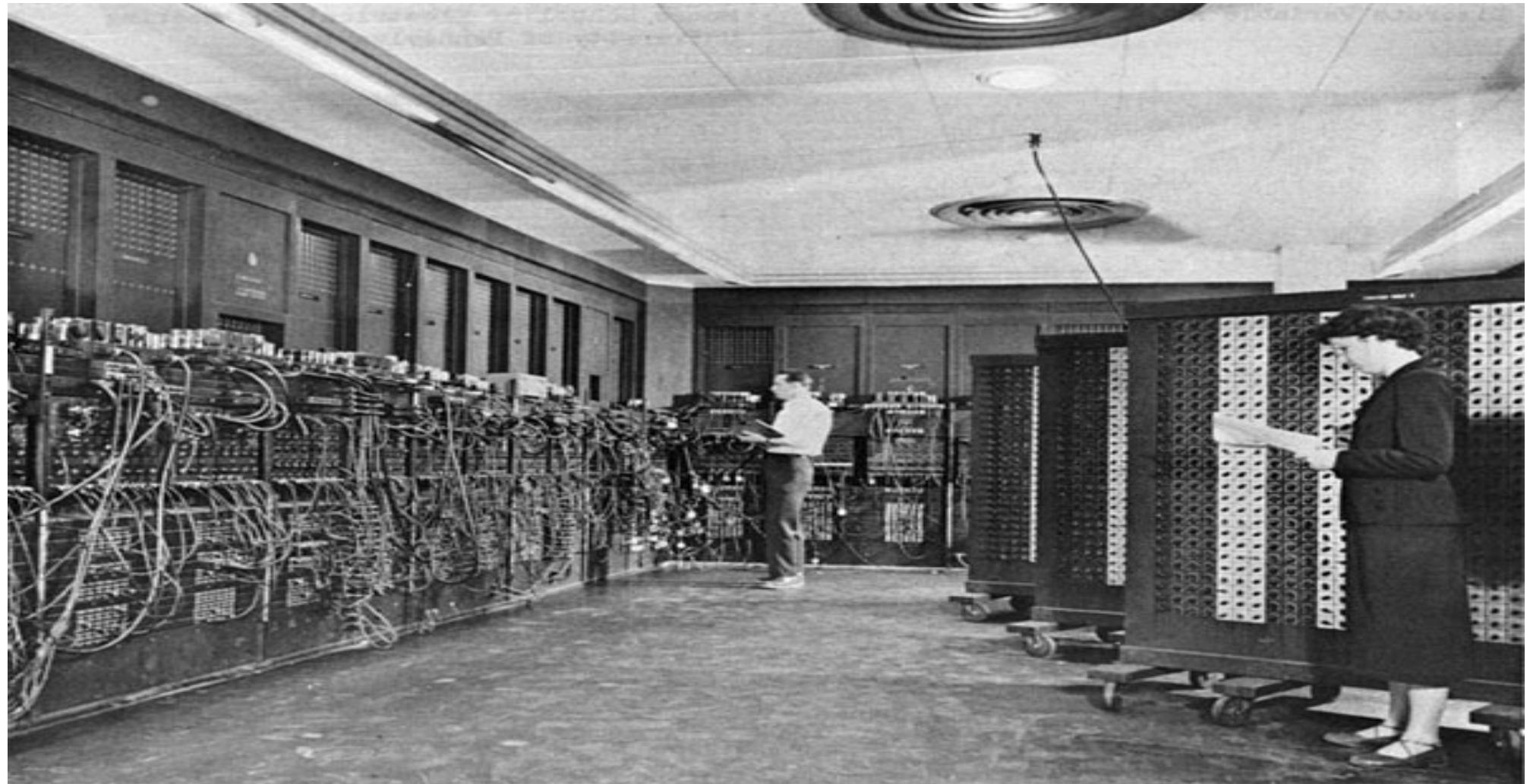
- Birkaç odayı doldurabilen
- Orta halli bir fabrikanın harcadığı elektriği harcayabilen
- Milyonlarca dolar fiyatı olan (1940'lı yıllarda)
- Sadece bir hesap makinesinin yaptığı işlemleri yapabilen
- Programlama için makine dili kullanan bir makine
- Detaylı Bilgi için:

<https://www.computerhope.com/issues/ch000984.htm>

# ENIAC

ENIAC , 1949  
University of  
Pennsylvania

J. Presper Eckert  
John Mauchly



# Makine Dili

İki sayının en büyük ortak bölenini hesaplayan makine dilindeki program

55	89	e5	53	83	ec	04	83	e4	f0	e8	31	00	00	00	89
c3	e8	2a	00	00	00	39	c3	74	10	8d	b6	00	00	00	00
39	c3	7e	13	29	c3	39	c3	75	f6	89	1c	24	e8	6e	00
00	00	8b	5d	fc	c9	c3	29	d8	eb	eb	90				

- makine dilini kullanarak program yazılmasının zorluğu
- programlar yazılrken hata oranın çok yüksek olması yeni bir dil arayışına yol açtı

# Assembly

İki sayının en büyük ortak bölenini hesaplayan assembly dilindeki program

pushl	%ebp	jle	D
movl	%esp, %ebp	subl	%eax, %ebx
pushl	%ebx	B:	cmpl %eax, %ebx
subl	\$4, %esp	jne	A
andl	\$-16, %esp	C:	movl %ebx, (%esp)
call	getint	call	putint
movl	%eax, %ebx	movl	-4(%ebp), %ebx
call	getint	leave	
cmpl	%eax, %ebx	ret	
je	C	D:	subl %ebx, %eax
A:	cmpl %eax, %ebx	jmp	B

Assembly dili yapılacak olan işlemlerin hatırlatıcı kısaltmalarla (*mnemonic abbreviations*) yazılmasını sağlayıp programın anlaşılabilir bir biçimde yazılmasına olanak sağlar.

# Derleyici (Assembler) nedir?

Assembly dilini makine diline programı çeviren sistem programlarına derleyici denir.

pushl	%ebp	jle	D
movl	%esp, %ebp	subl	%eax, %ebx
pushl	%ebx	B:	cmpl %eax, %ebx
subl	\$4, %esp	jne	A
andl	\$-16, %esp	C:	movl %ebx, (%esp)
call	getint	call	putint
movl	%eax, %ebx	movl	-4(%ebp), %ebx
call	getint	leave	
cmpl	%eax, %ebx	ret	
je	C	D:	subl %ebx, %eax
A:	cmpl %eax, %ebx	jmp	B



55	89	e5	53	83	ec	04	83	e4	f0	e8	31	00	00	00	89
c3	e8	2a	00	00	00	39	c3	74	10	8d	b6	00	00	00	00
39	c3	7e	13	29	c3	39	c3	75	f6	89	1c	24	e8	6e	00
00	00	8b	5d	fc	c9	c3	29	d8	eb	eb	90				

# Derleyici (Assembler) nedir?

Assembly dilini makine diline programı çeviren sistem programlarına derleyici denir.

pushl	%ebp	jle	D
movl	%esp, %ebp	subl	%eax, %ebx
pushl	%ebx	B:	cmpl %eax, %ebx
subl	\$4, %esp	jne	A
andl	\$-16, %esp	C:	movl %ebx, (%esp)
call	getint	call	putint
movl	%eax, %ebx	movl	-4(%ebp), %ebx
call	getint	leave	
cmpl	%eax, %ebx	ret	
je	C	D:	subl %ebx, %eax
A:	cmpl %eax, %ebx	jmp	B



PROBLEM

Programlama makine diline bağlı olarak devam etti ve her bir makine için kendine ait assembly dili kullanıldı.

# Problem: Derleyici (Assembler)

pushl	%ebp	jle	D
movl	%esp, %ebp	subl	%eax, %ebx
pushl	%ebx	B:	cmpl %eax, %ebx
subl	\$4, %esp	jne	A
andl	\$-16, %esp	C:	movl %ebx, (%esp)
call	getint	call	putint
movl	%eax, %ebx	movl	-4(%ebp), %ebx
call	getint	leave	
cmpl	%eax, %ebx	ret	
je	C	D:	subl %ebx, %eax
A:	cmpl %eax, %ebx	jmp	B



55	89	e5	53	83	ec	04	83	e4	f0	e8	31	00	00	00	89
c3	e8	2a	00	00	00	39	c3	74	10	8d	b6	00	00	00	00
39	c3	7e	13	29	c3	39	c3	75	f6	89	1c	24	e8	6e	00
00	00	8b	5d	fc	c9	c3	29	d8	eb	eb	90				

- Programlama makine diline bağlı olarak devam etti ve her bir makine için kendine ait assembly dili kullanıldı.
- Bilgisayarların gelişimi, donanımının karmaşıklaşmasıyla ve her bir makinenin kendi dilinin olması programlamayı çok zor bir hale getirdi ve bu durum programcılara kullanılan makineden bağımsız bir dilin gelişimine yol açtı.

# Programlama Alanları

- Bilimsel Uygulamalar
  - 1940'lı yıllar da geliştirilen ilk bilgisayarlar, ALGOL 60, FORTRAN
- İş ve Ticaret Uygulamaları (sayısal hesaplar ve veri işleme)
  - İş dünyasında kullanılan ilk yüksek seviyeli dil COBOL
- Yapay Zeka
  - LISP (McCarthy et al. 1965), Prolog (Clocksin , Mellish 2003) , C , Schema
- Sistem Programlama
  - IBM de geliştirilen ALGOL
  - UNIX dili C dili ile yazılmıştır
- Web
  - XHTML, PHP , Java, JavaScript

# Programlama Dillerinin Tarihsel Gelişimi

- 1950'de ilk yüksek seviyeli dil olan FORTRAN her makine için farklı assembly kod yazma problemini çözmek için geliştirildi.
- LISP (McCarthy et al. 1965)
- 1960 - 1970 li yıllarda structured programming ile birlikte
  - FORTRAN, COBOL ve BASIC gibi diller öne çıktı
  - goto yapısı , döngü ( while loop), switch yapısının gelişimine yol açtı.
- 1980 yıllarda nested block structure nesne tabanlı programların gelişimine yol açtı
  - ALGOL, PASCAL ve ADA; Smalltalk, C++ ve Eiffel

# Programlama Dillerinin Değerlendirilmesi

Karşılaştırma Kriterleri

Nitelikler

	Okunabilirlik (Readability)	Yazılabilirlik (Writability)	Güvenilirlik (Reliability)
Simplicity (Basitlik)	+	+	+
Orthogonality	+	+	+
Data types	+	+	+
Syntax design	+	+	+
Support for abstraction		+	+
Expressivity		+	+
Type checking			+
Exception handling			+
Restricted aliasing			+

# Programlama Dillerinin Çeşitliliği

- Özel amaçlı tasarım
    - Lisp sembolic ve karmaşık veri yapıları için uygundur.
    - Icon and Awk karakter işlemede kullanılmaktadır.
    - C düşük seviyeli sistem programlama da gerekmektedir.

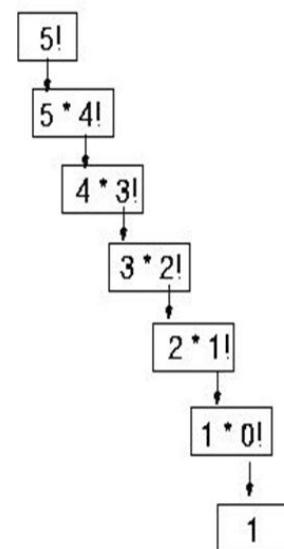


# Programlama Dillerinin Çeşitliliği

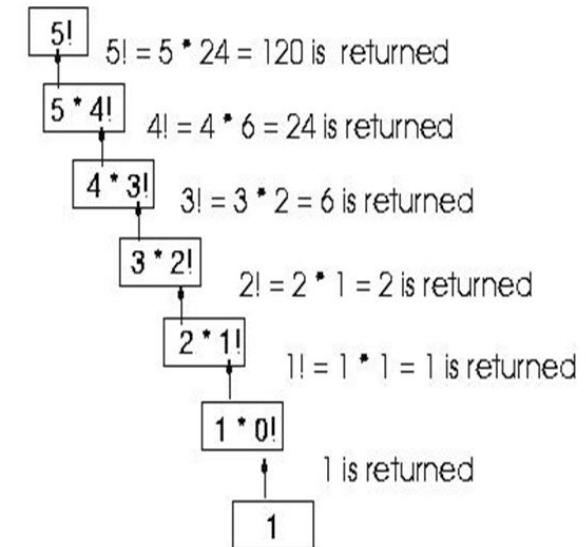
- Kişisel tercih
  - Kimi programcılar pointer'ları severken kimisi sevmez
  - kimisi recursive yapıları severken kimisi iteration dan hoşlanır.
  - Kişisel tercihler herkes tarafından benimsenip kullanılan bir dilin

- Iterative Solution

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n - 1) \cdot (n - 2) \cdots 3 \cdot 2 \cdot 1 & \text{if } n \geq 1 \end{cases}$$



Final value = 120

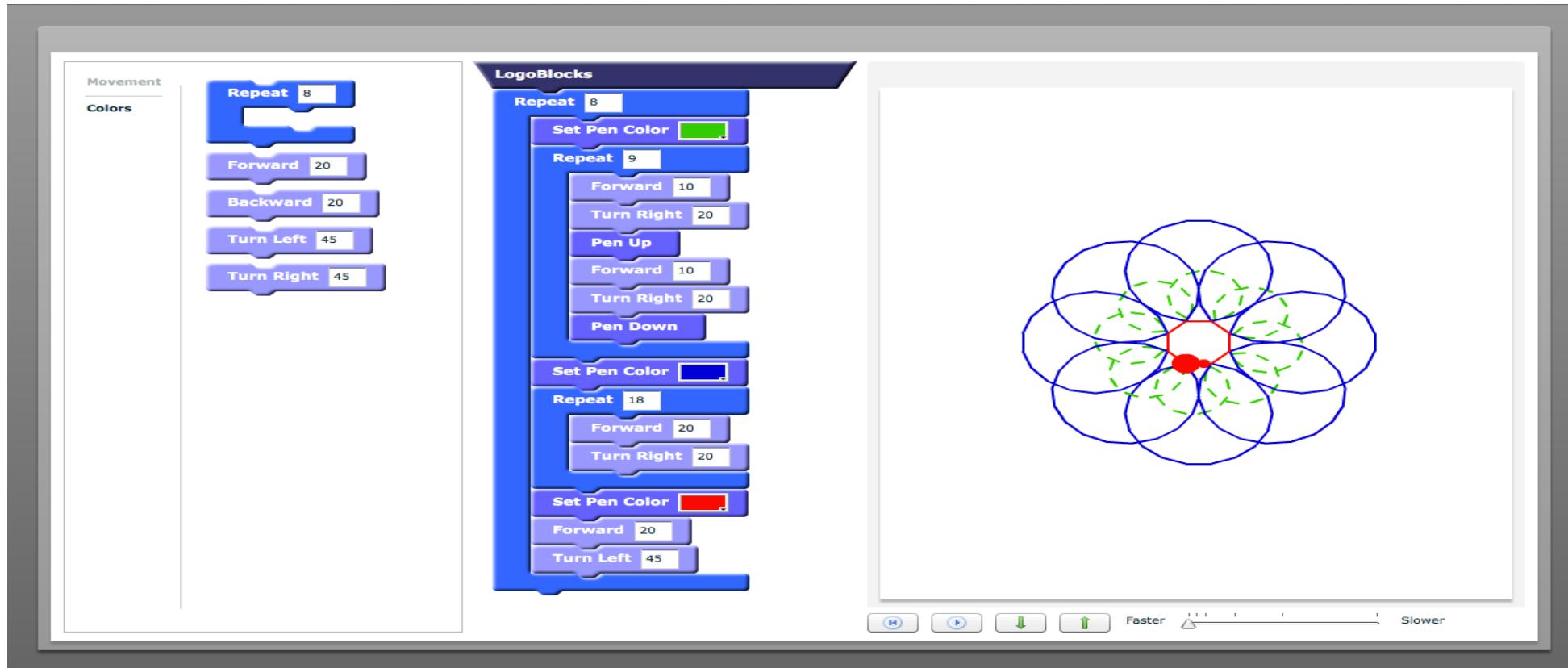


- Recursive Solution

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{factorial}(n - 1) & \text{if } n \geq 1 \end{cases}$$

# Programlama Dillerinin Çeşitliliği

- Yeni başlayanlara kolaylık sağlama: Logo dilinin kolay kullanımı 5 yaşındaki çocukların bile kodlamayı öğrenmesine yardımcı oluyor.



# Programlama Dillerinin Çeşitliliği

- ifade Gücü
  - Programlama dillerinin özellikleri kullanıcılarının kendi programlarını yazdıklarından kodun kısa anlaşılabilir olmasında etkisi büyük olduğundan, farklı programlama dilleri farklı kullanıcılar tarafından tercih edilebilir.

# Programlama Dillerinin Çeşitliliği

- **İfade Gücü**
  - Programlama dillerinin özellikleri kullanıcılarının kendi programlarını yazdıklarında kodun kısa anlaşılır olmasında etkisi büyük olduğundan, farklı programlama dilleri farklı kullanıcılar tarafından tercih edilebilir.
- **Programlamada ki kolaylık**
  - Öğrenilmesi ve programlaması kolay olan dillerin kullanımı artmıştır.
  - PASCAL dilini yazan Niklaus Wirth bütün üniversitelere PASCAL'ın ücretsiz kullanım izni vermiştir.
  - JAVA dili de aynı yolu izlemektedir.

# Programlama Dillerinin Çeşitliliği

- Standartlaştırma
  - Standartlaştırma işlemi bir dilin platformlar arasında kullanılmasını sağlayan en önemli faktördür.
  - Örneğin, bazı özelliklerini standart hale getiremeyen PASCAL'ın kullanımı 1980'li yıllarda itibaren bırakılmıştır.

# Programlama Dillerinin Çeşitliliği

- Açık kaynak
  - C dili 1970'li yıllarda Dennis Ritchie ve Ken Thompson tarafından Bell Labs 'da Unix ile beraber geliştirilmiştir.
  - C' nin standartlaştırılmasıyla birlikte kullanımı yaygınlaşmıştır.
  - Linux işletim sistemi, dünyaca çok yaygın olarak kullanılan, C ile yazılmıştır.

# Programlama Dillerinin Çeşitliliği

- Ekonomik destek
  - Bir dilin gelişmesi için güçlü bir sponsorun olması önemlidir.
  - Cobol ve Ada : ABD savunma bakanlığı destekliyor.
  - C# 'ı Microsoft destekliyor.

# Programlama Dillerinin Sınıflandırılması

Programlama dilleri hesaplama ve çalışma yöntemlerine göre sınıflandırılmaktadırlar iki ana sınıf altında incelenir:

## 1. Tarifli-Tanımlayıcı (Declarative Languages) Diller

- Bilgisayarın ne yapacağı tanımlanır.
- Yüksek seviyeli declarative dillerin gelişiminde düşük seviyeli programmanın detaylarını gizleme amacı yatkınlıkta.
- Bu diller ile programcılar çözülmlesi gereken problemin ana hatlarını belir geri kalan kısmı programlama dili tarafından yürütülür.

## 2. Emir-Komut (Imperative Languages) Dilleri

- Bilgisayarın işlemi nasıl yapacağı tanımlanır. Yapılacak işlem için adım adım kodlama yapılır. Performans amaçlı kullanılan dillerdir.

# Declarative Diller

## Tarifli-Tanımlayıcı Diller (Declarative Languages)

- Bilgisayarın ne yapacağı tanımlanır. Yüksek seviyeli declarative dillerin gelişiminde düşük seviyeli programlamanın detaylarını gizleme amacı yatmaktadır. Bu diller ile programcılar çözülmlesi gereken problemin ana hatlarını belir geri kalan kısmı programlama dili tarafından yürütülür.

### 1. Fonksiyonel

- Lisp, Haskel, ML, Scheme, Scala, Clojure
- Problemlerin çözümü recursive olarak yapılır.

	#1 Elixir
	#2 Scala
	#3 F#
	#4 Elm
	#5 Scheme
	#6 Clojure

# Declarative Diller

## Tarifli-Tanımlayıcı Diller (Declarative Languages)

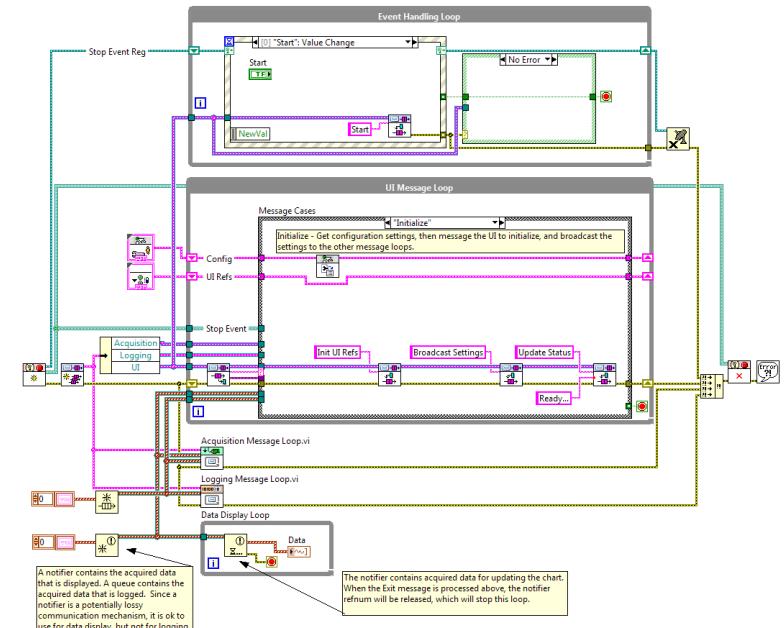
- Bilgisayarın ne yapacağı tanımlanır. Yüksek seviyeli declarative dillerin gelişiminde düşük seviyeli programlamanın detaylarını gizleme amacı yatmaktadır. Bu diller ile programcılar çözülmlesi gereken problemin ana hatlarını belir geri kalan kısmı programlama dili tarafından yürütülür.

### 1. Fonksiyonel

- Lisp, Haskel, ML, Scheme, Scala, Clojure
- Problemlerin çözümü recursive olarak yapılır

### 2. Veri Akışı

- Id, Val, LabView



# Declarative Diller

## Tarifli-Tanımlayıcı Diller (Declarative Languages)

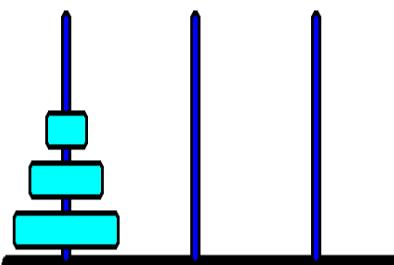
- Bilgisayarın ne yapacağı tanımlanır. Yüksek seviyeli declarative dillerin gelişiminde düşük seviyeli programlamanın detaylarını gizleme amacı yatmaktadır. Bu diller ile programcılar çözülmlesi gereken problemin ana hatlarını belir geri kalan kısmı programlama dili tarafından yürütülür.

1. Fonksiyonel

2. Veri akışı

3. Mantıksal ve kısıtlama tabanlı

- Prolog



```
hanoi(1, A, B, C, [A to B]).  
hanoi(N, A, B, C, Ms) :-  
    N > 1, R is N-1,  
    lemma(hanoi(R, A, C, B, M1)),  
    hanoi(N1, C, B, A, M2),  
    append(M1, [A to B|M2], Ms).
```

```
lemma(P) :- call(P),  
          asserta((P :- !)).
```

```
go(N, Pegs, Moves) :-  
    hanoi(N, A, B, C, Moves),  
    Pegs=[A, B, C].
```

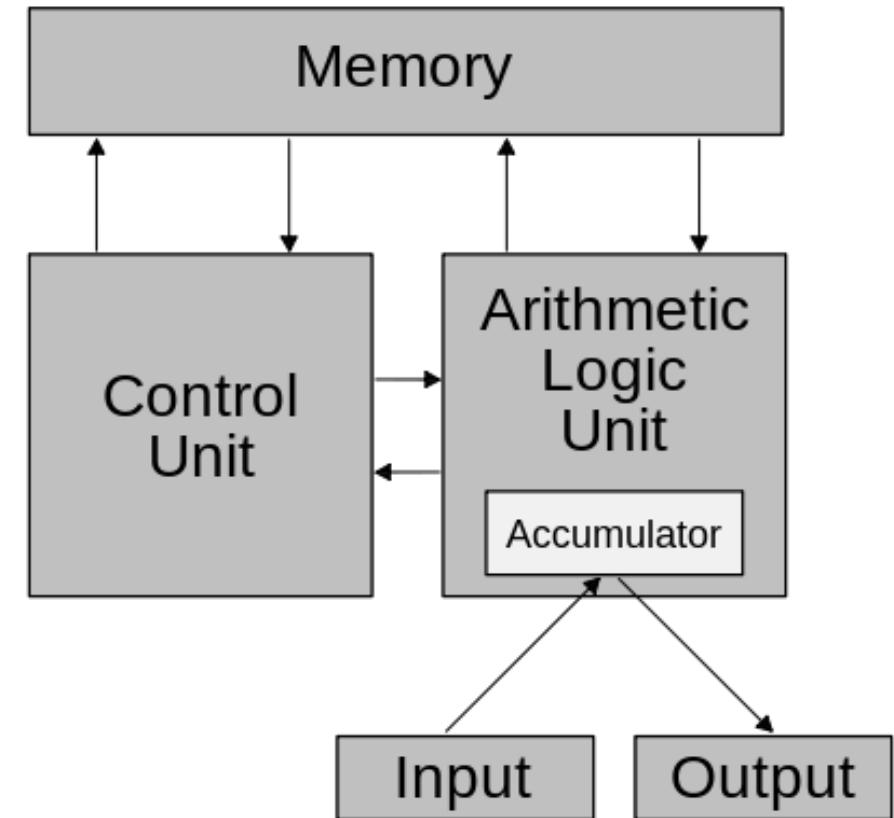
# İmperative Diller

- Emir-Komut (imperative Languages) Dilleri
  - Bilgisayarın işlemi nasıl yapacağı tanımlanır.
  - Yapılacak işlem için adım adım kodlama yapılır.
  - Performans amaçlı kullanılan dillerdir.
  - Von Neuman mimarisini kullanılır

# İmperative Diller

- Emir-Komut (imperative Languages) Dilleri
  - Bilgisayarın işlemi nasıl yapacağı tanımlanır.
  - Yapılacak işlem için adım adım kodlama yapılır.
  - Performans amaçlı kullanılan dillerdir.
  - Von Neuman mimarisini kullanılır

John von Neumann 'in stored program kavramının geliştirenlerin öncüsüdür. Donanımı kullanarak programın kaydedilmesi verilerin işlenmesi kaydedilmesi ve güncellenmesini sağlamışlardır.



[https://en.wikiquote.org/wiki/John\\_von\\_Neumann](https://en.wikiquote.org/wiki/John_von_Neumann)

# imperative Diller

## 1- Von Neuman

- C, Ada, Fortran
- İşlemler değişkenlerin sakladığı ve hafızadaki değerlerinin değişmesiyle gerçekleşmektedir.

# imperative Diller

## 2- Scripting

- Shell, Awk , Ruby, Perl, Python, PHP, JavaScript
- Von Neuman dillerinin bir alt sınıfıdır.
- Farklı dillerin iç içe kullanımına imkân sağlar.
- Program parçalarını birleştirmek amacıyla geliştirilmiştir.
- Hızlı bir biçimde uygulama geliştirme amacıyla kullanılmaktadır

# imperative Diller

## 3- Object-oriented

- Java, C++, Smalltalk, Eifel
- Simula67 temellerine dayanan nesne tabanlı programlama dilleri von neumann yapısını kullanmakla birlikle hesaplama ve hafıza kullanımında daha da yapısal ve dağıtıktır.

# OBEB Örneği

```
55 89 e5 53 83 ec 04 83 e4 f0 e8 31 00 00 00 89  
c3 e8 2a 00 00 00 39 c3 74 10 8d b6 00 00 00 00  
39 c3 7e 13 29 c3 39 c3 75 f6 89 1c 24 e8 6e 00  
00 00 8b 5d fc c9 c3 29 d8 eb eb 90
```

pushl %ebp	jle D
movl %esp, %ebp	subl %eax, %ebx
pushl %ebx	B: cmpl %eax, %ebx
subl \$4, %esp	jne A
andl \$-16, %esp	C: movl %ebx, (%esp)
call getint	call putint
movl %eax, %ebx	movl -4(%ebp), %ebx
call getint	leave
cmpl %eax, %ebx	ret
je C	D: subl %ebx, %eax
A: cmpl %eax, %ebx	jmp B

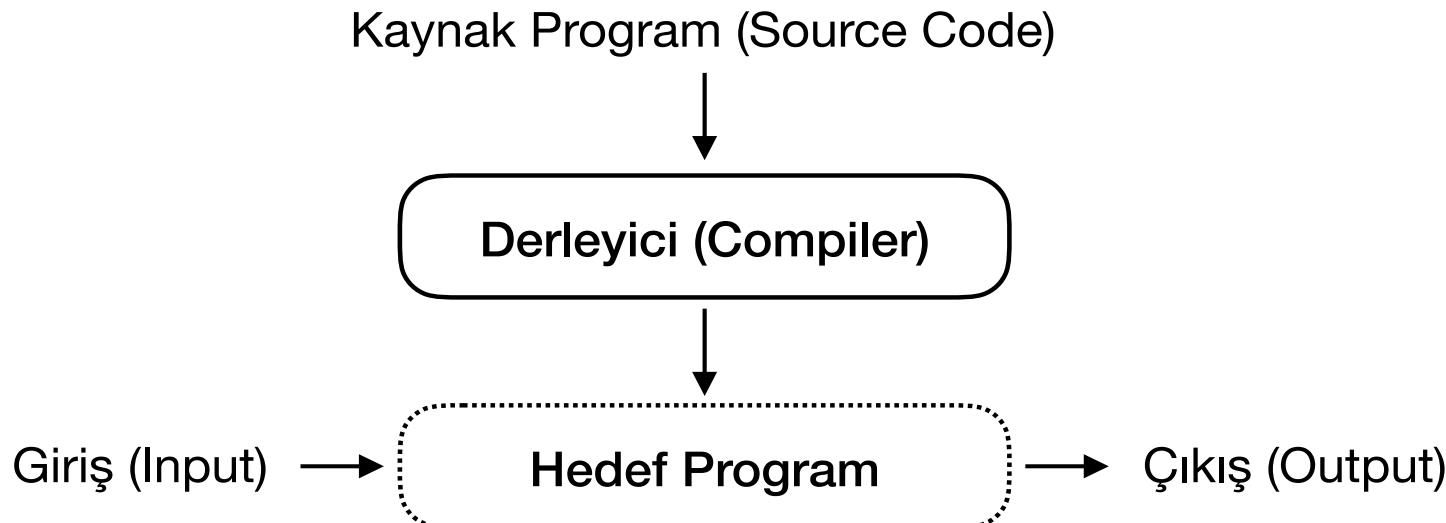
```
int gcd(int a, int b) { // C
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}

(define gcd ; Scheme
  (lambda (a b)
    (cond ((= a b) a)
          ((> a b) (gcd (- a b) b))
          (else (gcd (- b a) a)))))

gcd(A,B,G) :- A = B, G = A. % Prolog
gcd(A,B,G) :- A > B, C is A-B, gcd(C,B,G).
gcd(A,B,G) :- B > A, C is B-A, gcd(C,A,G).
```

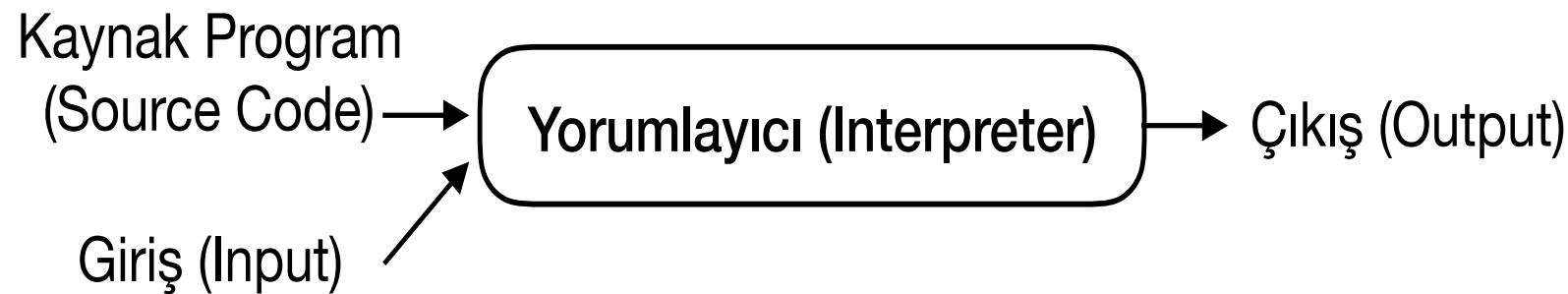
# Derleme (Compilation)

- Derleyici yüksek seviyeli dili düşük seviyeli makine diline çevirir.
- İşletim sistemi hedef programı girişleri alarak çalıştırır (*execution*).
- Derleyici derleme işlemi sırasındaki işlemi kontrol eder,
- Hedef program çalışma anındaki işlemleri kontrol eder.



# Yorumlama

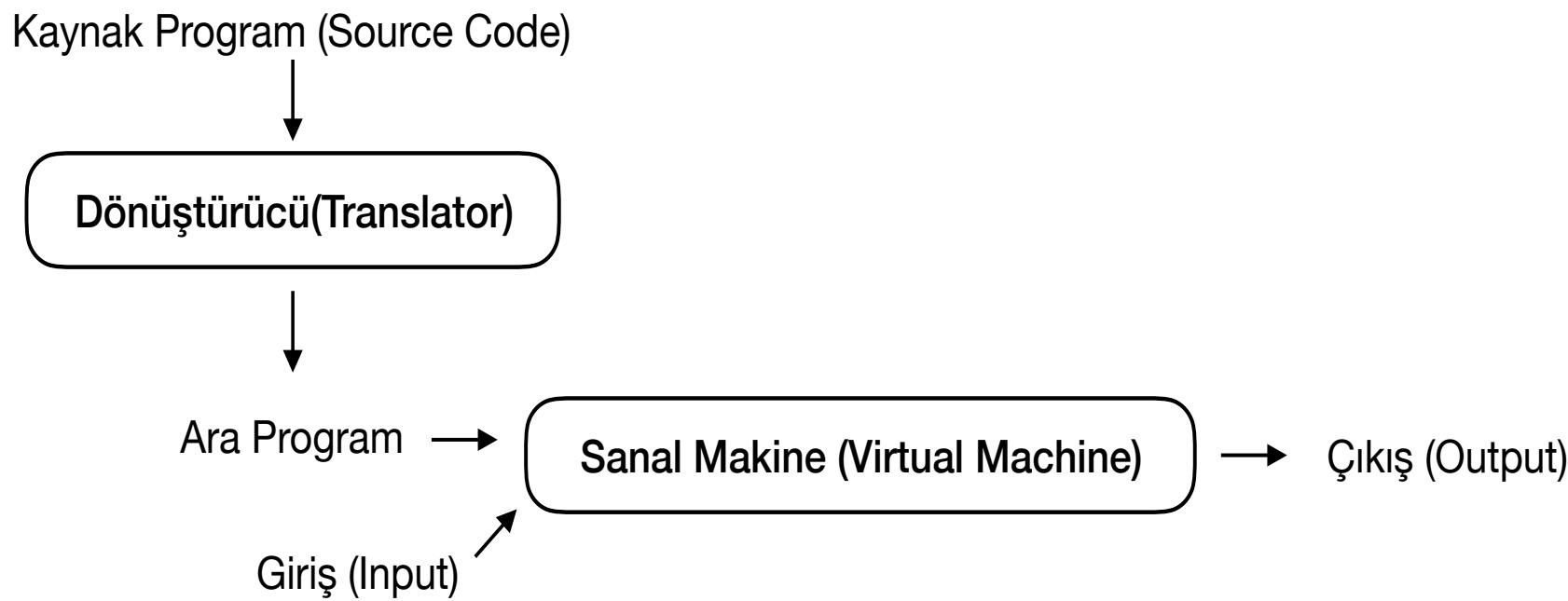
Derleyiciye alternatif olarak yüksek seviyeli diller için yorumlayıcılar bulunmaktadır.



Yorumlayıcının özellikleri :

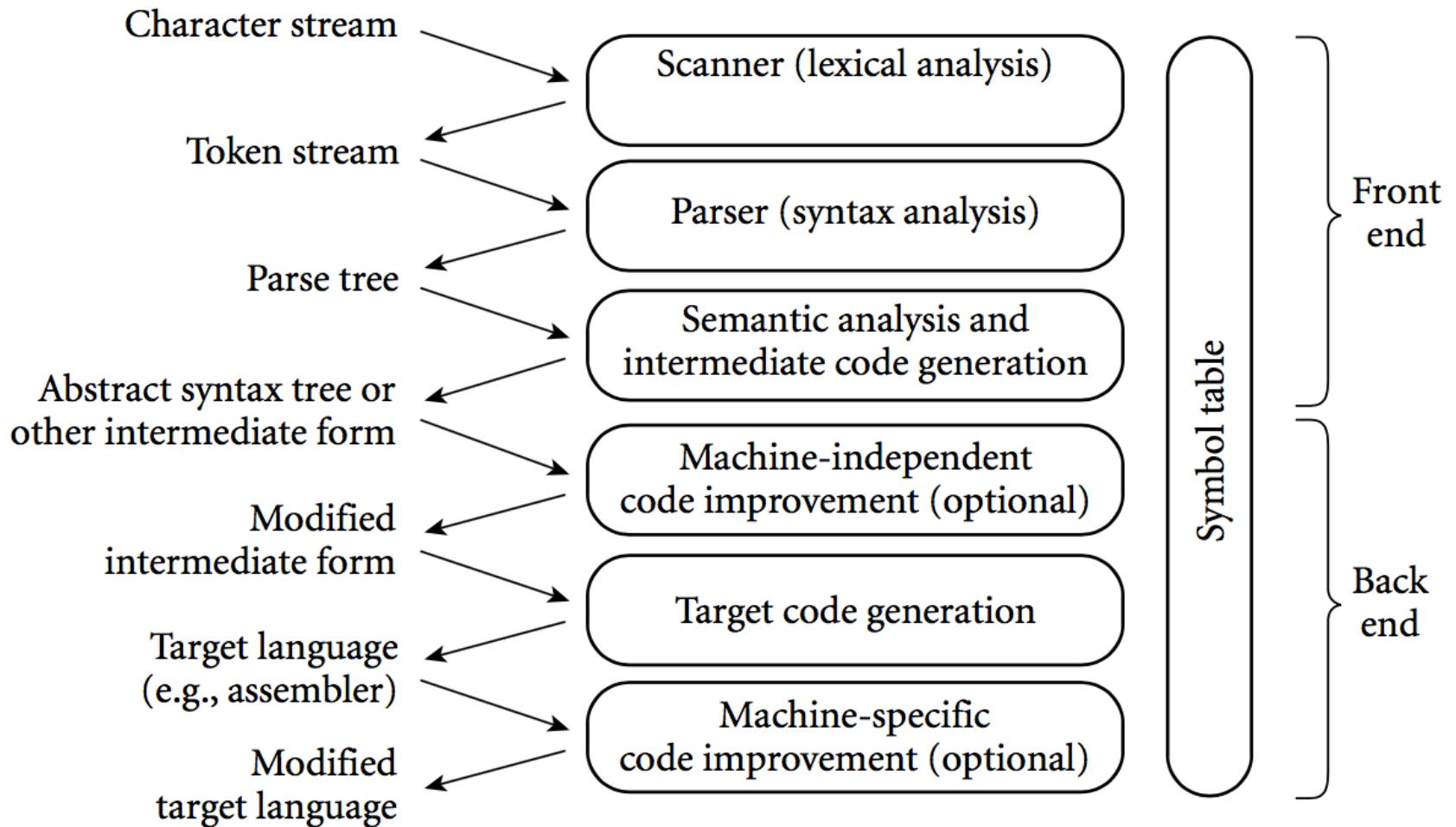
- Uygulamanın çalışma anını kontrol eder
- Kodu çalıştırırken sanal bir makine oluşturur ve yüksek seviyeli dili çalıştırır
- Yazılan kodu satır-satır çalıştırır
- Yorumlayıcı programdaki hataların bulunmasında esneklik ve kolaylık sağlar
- Yanlış ayıklayıcıların (*debugger*) kullanılmasına imkan sağlar.

# Derleme ve Yorumlama



- Bir programın son versiyonu bir defa derlenir ve sonrasında kullanılır.
- birçok programlama dilinde genellikle derleme ve yorumlama beraber yapılmaktadır. Java dilinde durum böyledir.

# Derleme Aşamaları



# Derleme Aşamaları

## 1- Tarama (Scanning)

'i', 'n', 't', ' ', 'm', 'a','i','n', '(',')',...

```
int main() {  
    int i = getInt(), j = getInt();  
    while (i != j) {  
        if (i > j) i = i - j;  
        else j = j - i;  
    }  
    putInt(i);  
}
```

# Derleme Aşamaları

2- Lexical Analysis: Token'lar oluşturulur.

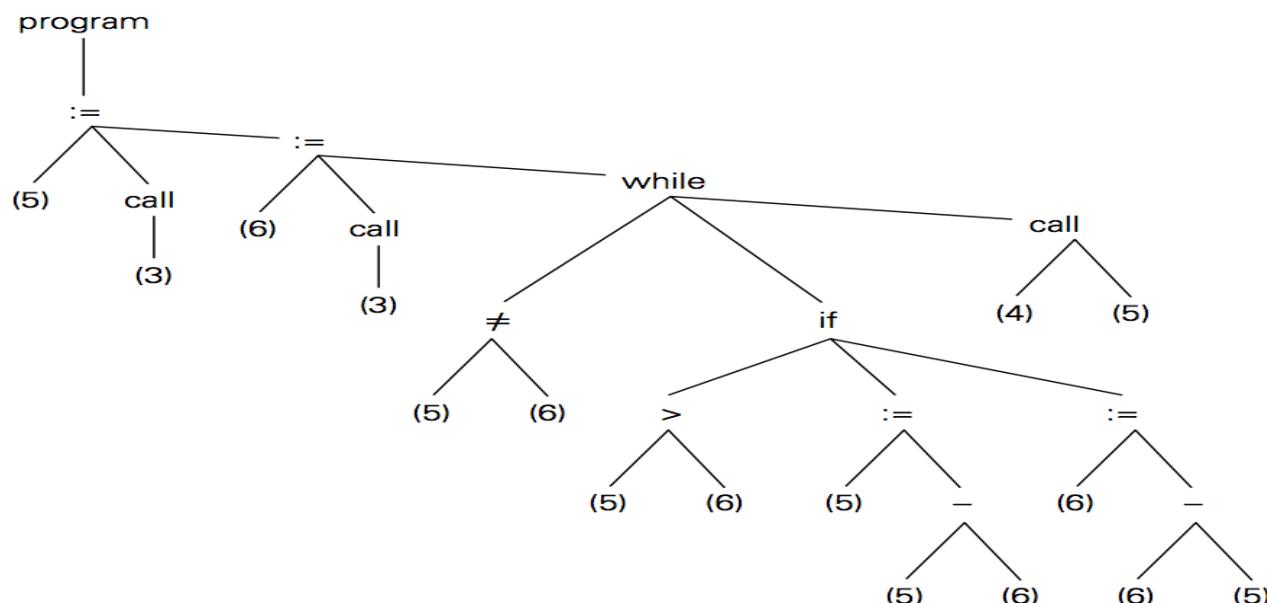
```
int main( ) { int i =  
getint( ), j = getint( )  
); while ( i != j )  
{ if ( i > j ) i  
= i - j ; else j =  
j - i ; } putint( i  
) ; }
```

```
int main() {  
    int i = getInt(), j = getInt();  
    while (i != j) {  
        if (i > j) i = i - j;  
        else j = j - i;  
    }  
    putInt(i);  
}
```

# Derleme Aşamaları

3- Parsing işlemi gerçekleştirilir

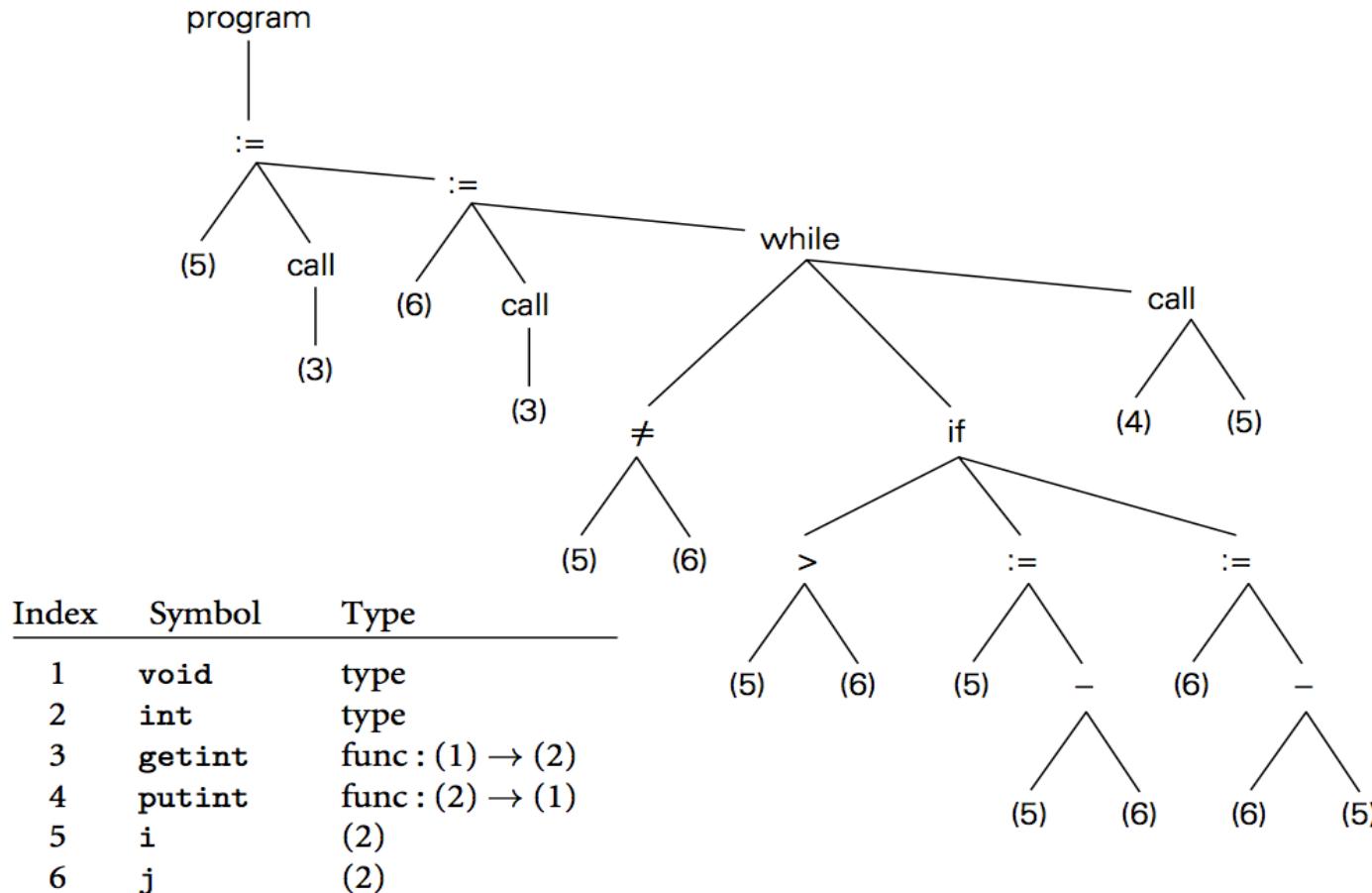
Tokenlar Yardımıyla programın manası bilinmeden  
ayrıştırma ağacı oluşturulur.



```
int main() {
    int i = getInt(), j = getInt();
    while (i != j) {
        if (i > j) i = i - j;
        else j = j - i;
    }
    putInt(i);
}
```

# Derleme Aşamaları

4- Anlamsal Analiz yapılır ve Sembol tablosu oluşturulur



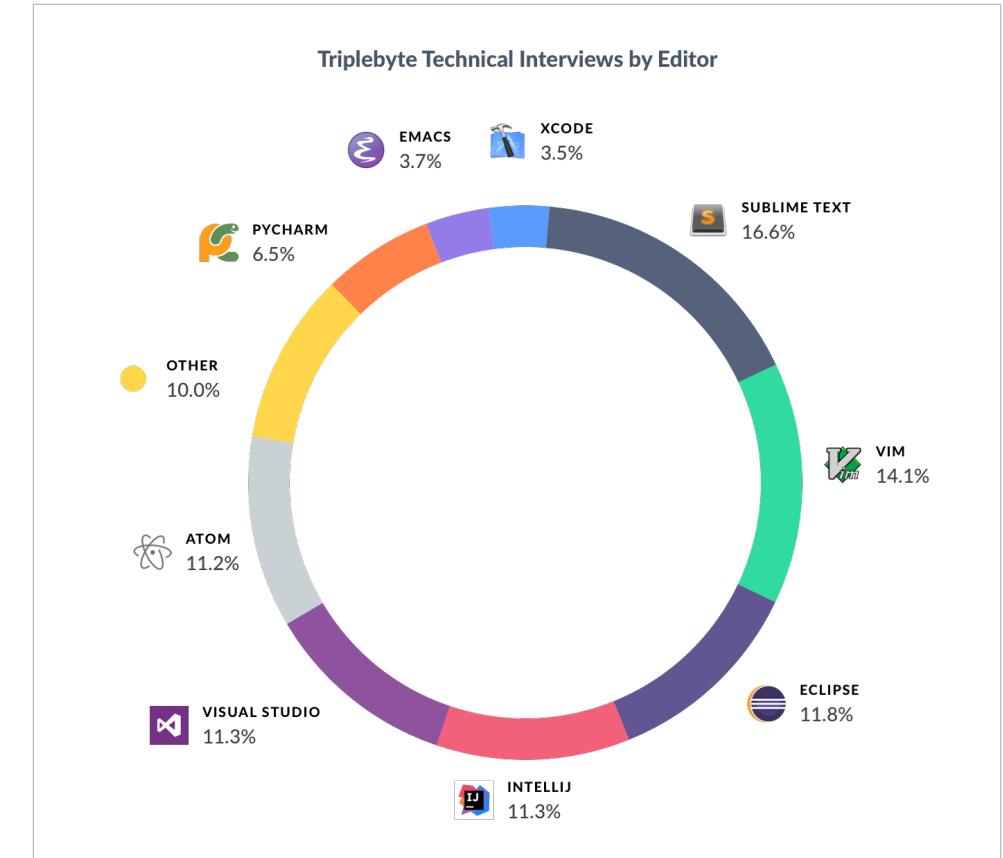
```
int main() {
    int i = getInt(), j = getInt();
    while (i != j) {
        if (i > j) i = i - j;
        else j = j - i;
    }
    putInt(i);
}
```

# Programlama Ortamı

- Text editörler
  - Atom
  - Sublime
  - Emacs
  - Vim
  - Notepad++

# Programlama Ortamı

- **IDE (Integrated Development Environments)**
  - Java
    - Eclipse, Netbeans, IntelliJ
  - Python
    - Pycharm, PyDev, Eclipse, Spyder
  - Ruby
    - Ruby mine , Emacs, Sublime
  - C++
    - Netbeans, Eclipse



[https://triplebyte.com/blog/technical-interview-performance-by-editor-os-language?imm\\_mid=0f6a21&cmp=em-prog-na-na-newsltr\\_20170923](https://triplebyte.com/blog/technical-interview-performance-by-editor-os-language?imm_mid=0f6a21&cmp=em-prog-na-na-newsltr_20170923)

# Programlama Ortamı

- Derleyiciler ve yorumlayıcılar diğer geliştirme araçlarından izole edilmiş olarak bulunmazlar.
  - Assembler
  - Debugger
  - Preprocessor
  - Linker

# Programlama Ortamı

- IDE'lerin sağlamış olduğu
  - yanlış ayıklayıcılar (debugger) kesme noktaları (breakpointler) ile kod içerisinde bulunan hatanın tespiti kolaylaştırılır
  - syntax'a göre yazım hatalarını da denetler.
  - Kod tekrar çalıştırıldığında yeni versiyonu hazırlanır

# Programlama Ortamı

- Ücretli
  - Microsoft Visual Studio
  - Apple'ın XCode

# Programlama Dilleri Trend 2017

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

[Edit Ranking](#) | [Add a Comparison](#) |

Language Types (click to hide)

Web Mobile Enterprise Embedded

Language Rank Types Spectrum Ranking

1.	Python		100.0
2.	C		99.7
3.	Java		99.4
4.	C++		97.2
5.	C#		88.6
6.	R		88.1
7.	JavaScript		85.5
8.	PHP		81.4
9.	Go		76.1
10.	Swift		75.3
11.	Arduino		73.0
12.	Ruby		72.4
13.	Assembly		72.1
14.	Scala		68.3
15.	Matlab		68.0

# Programlama Dilleri Trend 2017

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

Edit Ranking | Add a Comparison | [Twitter](#) [Facebook](#)

Language Types (click to hide)

Web Mobile Enterprise Embedded

Language Rank Types Spectrum Ranking

1. Python		100.0
2. C		99.7
3. Java		99.4
4. C++		97.2
5. C#		88.6
6. R		88.1
7. JavaScript		85.5
8. PHP		81.4
9. Go		76.1
10. Swift		75.3
11. Arduino		73.0
12. Ruby		72.4
13. Assembly		72.1
14. Scala		68.3
15. Matlab		68.0

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

Edit Ranking | Add a Comparison | [Twitter](#) [Facebook](#)

Language Types (click to hide)

Web Mobile Enterprise Embedded

Language Rank Types Trending Ranking

1. Python		100.0
2. C		98.4
3. C++		97.7
4. Java		97.0
5. Swift		88.6
6. JavaScript		87.2
7. Go		86.5
8. R		84.4
9. C#		80.2
10. Ruby		79.1
11. PHP		78.8
12. Assembly		75.0
13. Arduino		73.8
14. Shell		73.5
15. Scala		72.7

# Programlama Dilleri Trend 2017

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

[Edit Ranking](#) | [Add a Comparison](#) |

Language Types (click to hide)

Web Mobile Enterprise Embedded

Language Rank Types Spectrum Ranking

1. Python		100.0
2. C		99.7
3. Java		99.4
4. C++		97.2
5. C#		88.6
6. R		88.1
7. JavaScript		85.5
8. PHP		81.4
9. Go		76.1
10. Swift		75.3
11. Arduino		73.0
12. Ruby		72.4
13. Assembly		72.1
14. Scala		68.3
15. Matlab		68.0

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

[Edit Ranking](#) | [Add a Comparison](#) |

Language Types (click to hide)

Web Mobile Enterprise Embedded

Language Rank Types Trending Ranking

1. Python		100.0
2. C		98.4
3. C++		97.7
4. Java		97.0
5. Swift		88.6
6. JavaScript		87.2
7. Go		86.5
8. R		84.4
9. C#		80.2
10. Ruby		79.1
11. PHP		78.8
12. Assembly		75.0
13. Arduino		73.8
14. Shell		73.5
15. Scala		72.7

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum Trending Jobs Open Custom

[Edit Ranking](#) | [Add a Comparison](#) |

Language Types (click to hide)

Web Mobile Enterprise Embedded

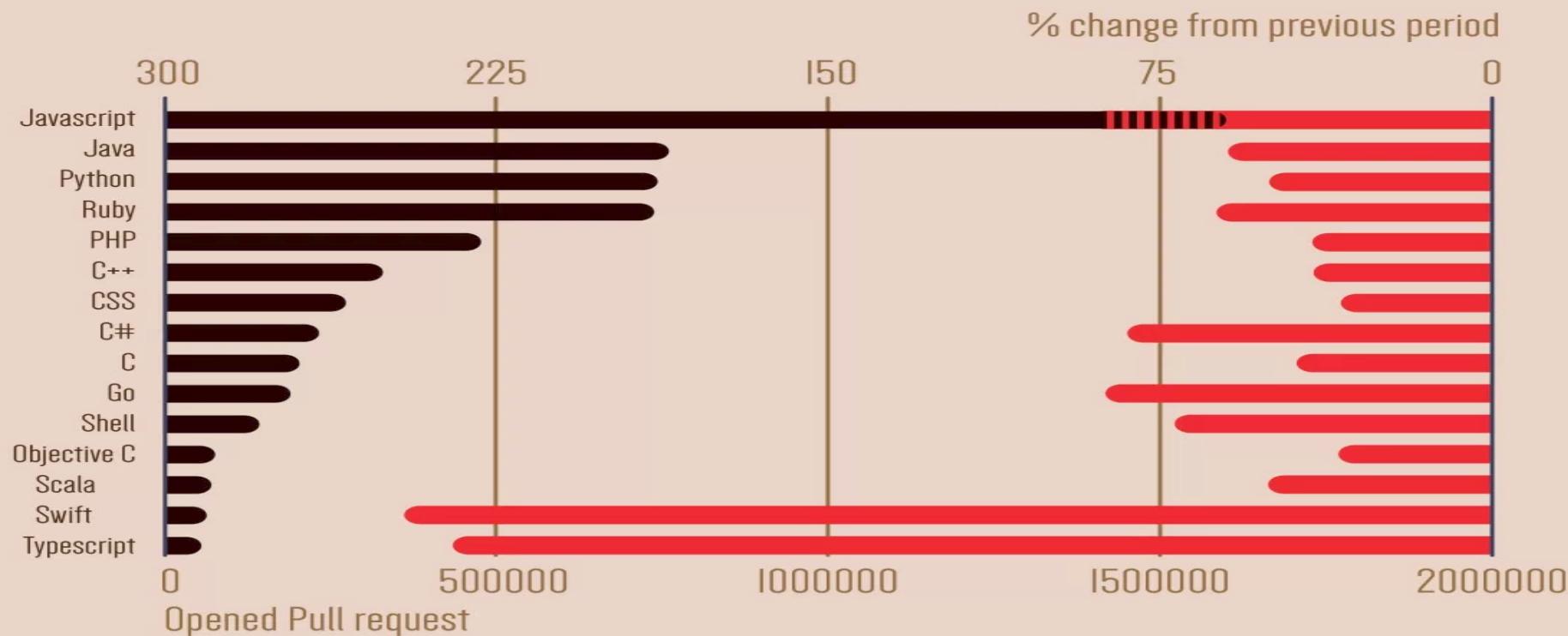
Language Rank Types Jobs Ranking

1. Java		100.0
2. C		99.3
3. Python		99.3
4. C++		92.6
5. JavaScript		90.2
6. C#		86.6
7. PHP		81.0
8. HTML		79.6
9. Ruby		77.2
10. Swift		77.2
11. Assembly		76.2
12. Shell		74.7
13. Scala		71.5
14. R		69.6
15. Perl		66.4

# Github En Populer PL 2017

hackerearth

## GITHUB MOST POPULAR LANGAUGES



# Bölüm Soruları

- Makine dilleri ile assembly dili arasındaki fark nedir? Assembly dilinin gelişimini sağlayan faktörler nelerdir?
- Çok çeşitli programlama dillerinin ortaya çıkışındaki faktörler nelerdir?
- Declarative programlama dillerinin alt sınıfları nelerdir?
- Imperative programlama dillerinin alt sınıfları nelerdir?
- Derleyici nedir? Yorumlayıcı nedir?
- Derleme ve yorumlama arasındaki fark nedir?
- Derleyici ve önişlemci arasındaki fark nedir?
- Programlama ortamlarında kullanılan IDE'lerin özellikleri nelerdir?
- Derleme aşamalarını nelerdir?