

# Programlama Dilleri (315)

---

**Dr. Öğr. Üyesi Ahmet Arif AYDIN**

Syntax ve Semantics Kavramları

# Söz Dizimi (Syntax)

Bir programlama dilinin

- ifadeleri (expressions)
- deyimleri (statement)
- program birimleri (program unit)
- yazım biçimi
- yazım kuralları



**SYNTAX**

# Söz Dizimi (Syntax)

- C ++
  - ifadelerin ardından noktalı virgül (;) kullanılması
- Java
  - Fonksiyon tanımlama

```
public Integer fonksiyon()
{
    return değer;
}
```

```
#include <cstdio>
#include <direct.h>

int main() {
    remove( "input.txt" );
    remove( "/input.txt" );
    _rmdir( "docs" );
    _rmdir( "/docs" );

    return 0;
}
```

# Söz Dizimi (Syntax)

The screenshot shows the Sublime Text editor interface. The title bar reads "Sublime Text" and the active file is "aydin.py". The code editor displays the following Python script:

```
1 #Author: Ahmet Arif Aydin
2 #Date : March, 2013
3
4 import re
5 import sys
6
7 # reads from file and finds all the lines which have "ResponseReader:314" text in
8 def read(fname):
9     list=[]
10    with open(fname) as f:
11        for line in f:
12            twt=[]
13            match=re.search("ResponseReader:314",line)
14            if match:
15                reader=line.strip()
16                reader1=reader.split(',')
17                list.append(reader1)
18
19    return list
20
21 # standard calls the main() function.
22 if __name__ == '__main__':
23
24    catalina=read(sys.argv[1])
25
26    print len(catalina)
27    print catalina[1]
```

The right-hand sidebar lists various programming languages and file types, with "Python" selected. The status bar at the bottom left indicates "Line 26 Column 22".

- ActionScript
- AppleScript
- ASP
- Batch File
- C
- C#
- C++
- Clojure
- CSS
- D
- Diff
- Erlang
- Go
- Graphviz (DOT)
- Groovy
- Haskell
- HTML
- Java
- JavaScript
- LaTeX
- Lisp
- Lua
- Makefile
- Markdown
- MATLAB
- Objective-C
- OCaml
- Pascal
- Perl
- PHP
- Plain Text
- ✓ Python**
- R
- Rails
- Regular Expression
- reStructuredText
- Ruby
- Rust
- Scala
- Shell Script (Bash)
- SQL
- TCL
- Textile
- XML
- YAML

# Anlam (Semantics)

---

Programlama dilinde bulunan ifadelerin, deyimlerin ve program birimlerinin manası



## SEMANTICS

Kullanılan yapının *nasıl bir sonuç vereceğini* tanımlar

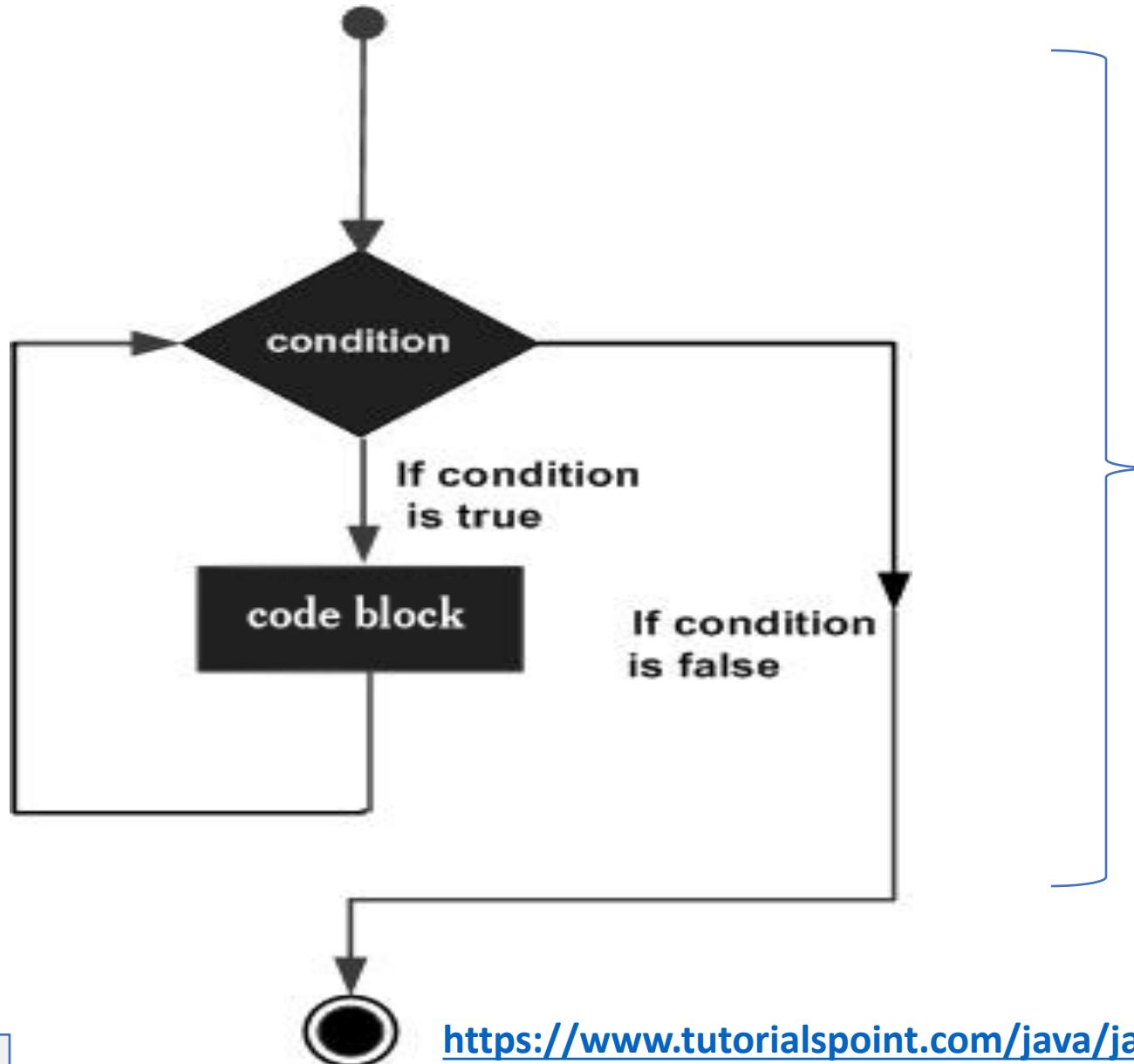
# While Döngüsü

```
While (şart)
{
    ....
    yapılacak işlemler
    ....
}
```



while döngüsünün nasıl yazılması  
gerektiğini **syntax** tanımlar

# While Döngüsü



**Semantics:** Eğer şart doğru ise işlemleri gerçekleştir. Değilse while döngüsünden çıkış ve sonraki satıra geç

*İyi bir program için tavsiye edilen semantik ve syntax'in uyumlu bir biçimde çalışmasıdır.*

# Syntax'i Tanımlama

---

- Doğal diller ve programlama dilleri kullanılan alfabetin karakter setlerinden oluşmaktadır.
- Her dilin kendine ait özellikleri ve söz dizimi kuralları (syntax) bulunmaktadır.
- Dilin kuralları yazılan karakterlerin anlamını (semantics) tanımlamaktadır.

# Syntax'i Tanımlama

---

- Doğal diller ve programlama dilleri kullanılan alfabetin karakter setlerinden oluşmaktadır.
- Her dilin kendine ait özellikleri ve söz dizimi kuralları (syntax) bulunmaktadır.
- Dilin kuralları yazılan karakterlerin anlamını (semantics) tanımlamaktadır.

03.09.1999

# Syntax'i Tanımlama

- Doğal diller ve programlama dilleri kullanılan alfabetin karakter setlerinden oluşmaktadır.
- Her dilin kendine ait özellikleri ve söz dizimi kuralları (syntax) bulunmaktadır.
- Dilin kuralları yazılan karakterlerin anlamını (semantics) tanımlamaktadır.

Türkiye'de  
3 Eylül 1999

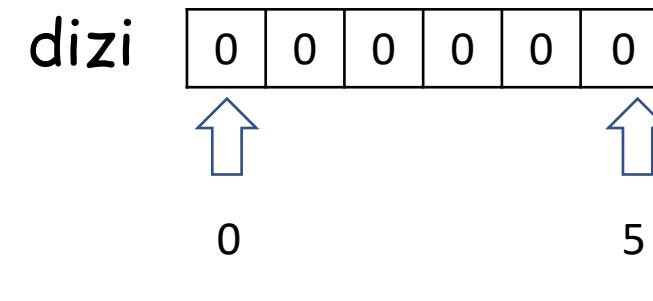


03.09.1999



ABD'de  
9 Mart 1999

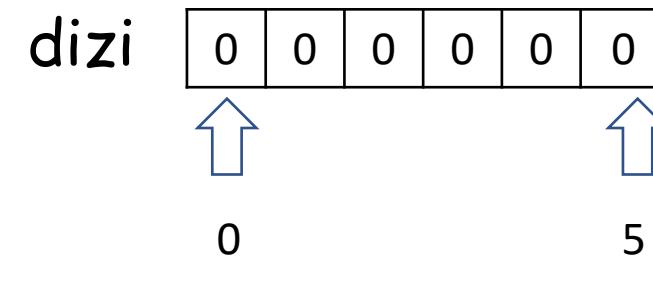
# Örnek: Syntax



```
4 public static void main(String args[]){
5
6     int [] dizi = new int[6];
7
8     int k=5;
9
10    while (k>1)
11        {
12
13            dizi[k]=k--;
14        }
15
16    }
```



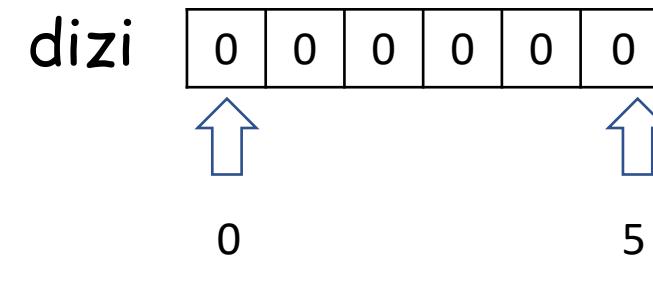
# Örnek: Syntax



```
4 public static void main(String args[]){
5
6     int [] dizi = new int[6];
7
8     int k=5;
9
10    while (k>1)
11        {
12
13            dizi[k]=k--;
14        }
15
16    }
```

0	0	2	3	4	5
---	---	---	---	---	---

# Örnek: Syntax



```
4 public static void main(String args[]){
5     int [] dizi = new int[6];
6
7     int k=5;
8
9     while (k>1)
10    {
11        dizi[k]=k--;
12    }
13}
```

0	0	2	3	4	5
---	---	---	---	---	---

```
4 public static void main(String args[]){
5     int [] dizi = new int[6];
6     int k=5;
7
8     while (k>1)
9     {
10        dizi[k]=--k;
11    }
12}
13
```



# Örnek: Syntax

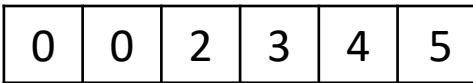
dizi

0	0	0	0	0	0
---	---	---	---	---	---

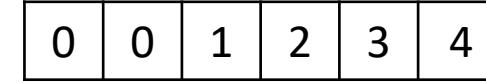
0                    5



```
4 public static void main(String args[]){
5
6     int [] dizi = new int[6];
7
8     int k=5;
9
10    while (k>1)
11        {
12
13        dizi[k]=k--;
14    }
15
16 }
```



```
4- public static void main(String args[]){
5     int [] dizi = new int[6];
6     int k=5;
7
8     while (k>1)
9     {
10        dizi[k]=--k;
11    }
12
13 }
```



# Lexeme ve Token

---

Programlama dilinde kod yazılrken

Her bir sözcük (lexeme) olarak tanımlanır

Anlamsal olan en küçük birime *token* denir.

Lexeme'ler token içerisinde kategorize edilir.

# Lexeme ve Token

not= ödev\*0.3 + vize\*0.3 + final\*0.4

Lexeme	Token
not, ödev, vize, final	Tanımlayıcı (identifier)
=	eşittir
*	çarpma işlemi
+	toplama işlemi
;	noktalı virgül

# Dillerin Formal Tanımlayıcıları

---

Dillerin tanımlanması için genel olarak iki yol bulunmaktadır:

1. Dil Tanıycıları (Language Recognizers)
2. Dil Üreticileri (Language Generators)

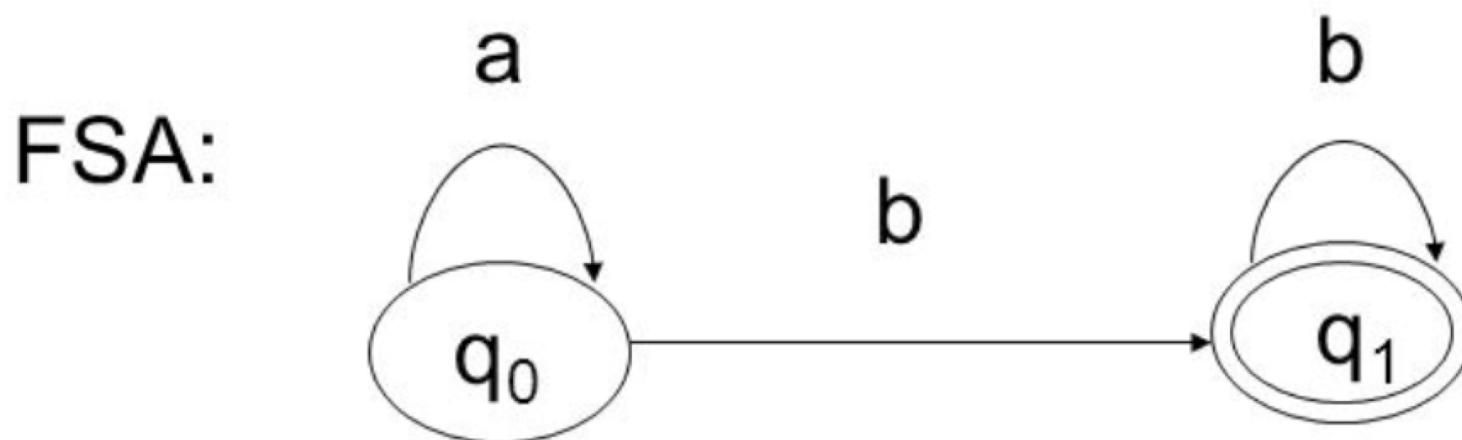
# 1-Dil Tanıycıları (Language Recognizers)

---

- Giriş olarak verilen bir karakterler kümesinin bir dilde olup olmadığını kontrol eder.
  - $L$  : bir dil
  - $\Sigma$  : Alfabe
  - $R$  : tanımlayıcı cihaz

# 1-Dil Tanıycıları (Language Recognizers): FSA

- FSA : Finite State Automata (Sonlu Durum Otomata)

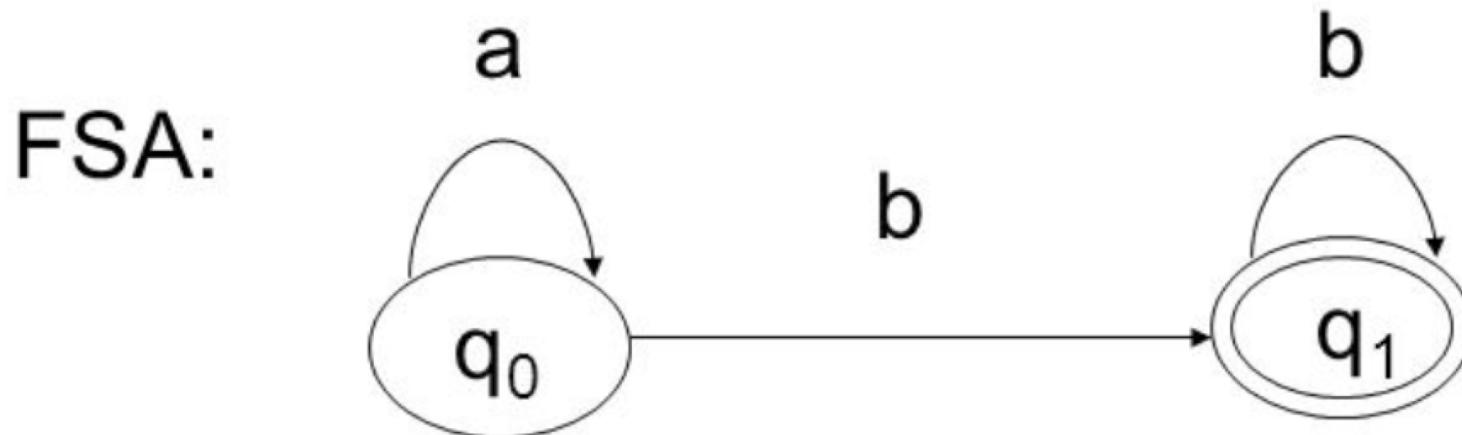


Regular language: {b, ab, bb, aab, abb, ...}

# 1 - Dil Tanıycıları (Language Recognizers): FSA

- FSA : Finite State Automata (Sonlu Durum Otomata)

$a^* b^+$   
 $a^n b^m \quad n \geq 0 \quad m > 0$



Regular language: {b, ab, bb, aab, abb, ...}

# 2-Dil Üreticileri (Language Generator)

---

- Bir dilde bulunabilecek cümleleri üretebilen cihazlardır.
- Dil tanıincinnarla beraber çalışmaktadır.

1. Context-free Grammar (CFG)
2. Backus-Naur Form (BNF)

# Context-free Grammar



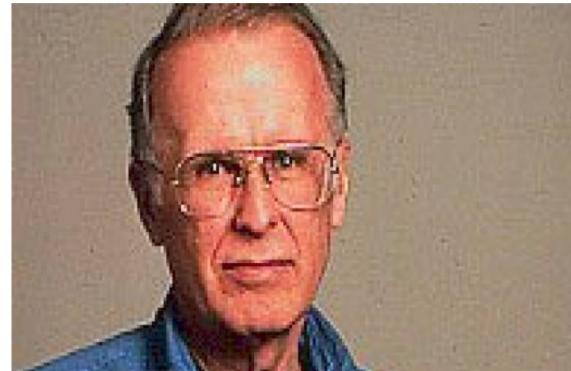
Noam Chomsky  
American linguist

- Dil bilimci olan Chomsky 1956 yılında iki gramer sınıfı geliştirdi:
  - Context-free gramer
  - Regular gramer

Chomsky geliştirdiği sınıfların programlama dillerine uygulanabileceğini bilmiyordu. Ancak daha sonra Context-free sınıfı ile de bütün programlama dillerinin syntax'ının tanımlanabileceği anlaşıldı.

# Backus-Naur Form (BNF)

- 1959 yılında ALGOL 58 ile beraber John Backus programlama dilleri syntax için yeni bir biçimsel gösterim (notation) ortaya konuldu.



John  
Backus

- 1960 yılında ALGOL 60 ile beraber Peter Naur, John Backus tarafından sunulan gösterimde bazı değişiklikler yaparak geliştirdi ve bu form Backus-Naur Form (BNF) olarak anılmaya başlandı.



Peter  
Naur

# Meta language

---

- Bir dili tanımlamak için kullanılan başka bir dile **üst-dil** (metalanguage) denir.
- Programlama dillerinin metalanguage'i BNF dir.
- BNF syntaxı tanımlamak için soyut ifadeler kullanmaktadır.
- BNF context-free grammer için kural oluşturmayı sağlar

# Örnek: Atama işlemi

**<atama>** → **<değişken>** = **<ifade>**



gerçekleştirilecek  
olan işlemin adını  
tanımlar



<atama> işleminin  
nasıl yapılacağını  
göstermektedir

lexeme, token ve  
referanslar  
bulunmaktadır.

# Örnek: Atama işlemi

---

Bu yapıya kural (rule) denir.

`<atama> → <değişken> = <ifade>`

Bu kural ile atama soyut tanımı, değişken, eşittir ve ifade ile belirlenmektedir.

# Backus-Naur Form (BNF): İF Yapısı

---

`<if-yapısı> → if ( <karşılaştırma> ) <işlem>`

`< if-yapısı > → if ( < karşılaştırma > ) < işlem > else < işlem >`

# Backus-Naur Form (BNF): İF Yapısı

<if-yapısı> → if ( <karşılaştırma> ) <işlem>

< if-yapısı > → if ( < karşılaştırma > ) < işlem > else < işlem >

< if-yapısı > → if ( < karşılaştırma > ) < işlem >  
| if ( < karşılaştırma > ) < işlem > else < işlem >

Bu yapı if yapısının birden fazla formunun olduğunu tanımlar.

Birden fazla kuralı birleştirmek için **OR** | simbolu kullanılır.

# Backus-Naur Form (BNF): Recursion

BNF de sıralı listelerin veya ardışık yapıları (1, 2, 3,.....)

tanımlamak için öz yineleme (recursion) kullanılır.

```
<liste> → tanımlayıcı  
          | tanımlayıcı , <liste>
```

# Backus-Naur Form (BNF)

- BNF de bulunan soyut tanımlamalar **Non-terminal symbols (NT)** olarak ifade edilir.
- Kurallar da bulunan Lexeme ve token **Terminal (T)** olarak adlandırılır.
- BNF (veya Gramer ) kurallar topluluğu olarak ifade edilir.

**<atama> → <değişken> = <ifade>**

<program> → başla < ifade-listesi > bitir

<ifade-listesi> → <atama>

| < atama > ; < ifade-listesi >

< atama > → <değişken> = <ifade>

< değişken > → X | Y | Z

< ifade > → < değişken > + < değişken >

| < değişken > – < değişken >

| < değişken >

# Gramer ve Türetme

<program> → başla < ifade-listesi > bitir

<ifade-listesi> → <atama>

| < atama > ; < ifade-listesi >

< atama > → <değişken> = <ifade>

< değişken > → X | Y | Z

< ifade > → < değişken > + < değişken >

| < değişken > – < değişken >

| < değişken >

- => sembolü türetir diye okunur.
- Her bir aşamada NT (Non-terminal) olan semboller Terminal olanlarla değiştirilerek türetme gerçekleştirilir.
- Türetilen her bir string sentential form olarak tanımlanır.
- Bütün NT semboller T olanlarla değiştirilinceye kadar işlem devam eder.
- İşlem soldan sağa doğru yapıldığı için leftmost derivations olarak adlandırılır.

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<program>** =>

< ifade-listesi > → < atama >

| < atama > ; < ifade-listesi >

< atama > → < değişken > = < ifade >

< değişken > → X | Y | Z

< ifade > → < değişken > + < değişken >

| < değişken > - < değişken >

| < değişken >

# Gramer ve Türetme

$\langle \text{program} \rangle \rightarrow \text{başla } \langle \text{ifade-listesi} \rangle \text{ bitir}$

$\langle \text{program} \rangle \Rightarrow \text{başla } \langle \text{ifade-listesi} \rangle \text{ bitir}$

$\langle \text{ifade-listesi} \rangle \rightarrow \langle \text{atama} \rangle$

|  $\langle \text{atama} \rangle ; \langle \text{ifade-listesi} \rangle$

$\langle \text{atama} \rangle \rightarrow \langle \text{değişken} \rangle = \langle \text{ifade} \rangle$

$\langle \text{değişken} \rangle \rightarrow X | Y | Z$

$\langle \text{ifade} \rangle \rightarrow \langle \text{değişken} \rangle + \langle \text{değişken} \rangle$

|  $\langle \text{değişken} \rangle - \langle \text{değişken} \rangle$

|  $\langle \text{değişken} \rangle$

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

< atama > ; < ifade-listesi >

**< atama >** → <değişken> = <ifade>

**< değişken >** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

**<program>** => başla < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| < atama > ; < ifade-listesi >

**< atama >** → <değişken> = <ifade>

**< değişken >** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

başla < atama >; < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <**değişken**> = <ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

başla <atama> < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| < atama > ; < ifade-listesi >

**< atama >** → <değişken> = <ifade>

**< değişken >** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

<atama> → <değişken> = < ifade>

**<değişken> → X | Y | Z**

< ifade > → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < **değişken** > = < ifade > ; < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi> => başla X = <ifade> ; <ifade-listesi> bitir

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

=> başla X = <değişken> + <değişken> ; <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

=> başla X = <değişken> + <değişken> ; <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = < ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = <değişken> + <değişken> ; < ifade-listesi > bitir

=> başla X = Y + <değişken> ; < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = < değişken > + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + Z ; < ifade-listesi > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

=> başla X = <değişken> + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + Z <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

<atama> → <değişken> = <ifade>

<değişken> → X | Y | Z

<ifade> → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

=> başla X = <değişken> + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + Z ; <ifade-listesi> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = < değişken > + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + Z ; < ifade-listesi > bitir

=> başla X = Y + Z ; < atama > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = < ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = < değişken > + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + Z ; < ifade-listesi > bitir

=> başla X = Y + Z ; < atama > bitir

=> başla X = Y + Z ; < değişken > = < ifade > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; <ifade-listesi>

**<atama>** → <değişken> = <ifade>

**<değişken>** → X | Y | Z

**<ifade>** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla <ifade-listesi> bitir

=> başla <atama> ; <ifade-listesi> bitir

=> başla <değişken> = <ifade> ; <ifade-listesi> bitir

=> başla X = <ifade> ; <ifade-listesi> bitir

=> başla X = <değişken> + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + <değişken> ; <ifade-listesi> bitir

=> başla X = Y + Z ; <ifade-listesi> bitir

⇒ başla X = Y + Z ; <atama> bitir

⇒ başla X = Y + Z ; <değişken> = <ifade> bitir

=> başla X = Y + Z ; Y = <ifade> bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = < ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = < değişken > + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + Z ; < ifade-listesi > bitir

⇒ başla X = Y + Z ; < atama > bitir

⇒ başla X = Y + Z ; < değişken > = < ifade > bitir

=> başla X = Y + Z ; Y = < ifade > bitir

# Gramer ve Türetme

**<program>** → başla < ifade-listesi > bitir

**<ifade-listesi>** → <atama>

| <atama> ; < ifade-listesi >

**<atama>** → <değişken> = < ifade>

**<değişken>** → X | Y | Z

**< ifade >** → <değişken> + <değişken>

| <değişken> - <değişken>

| <değişken>

<program> => başla < ifade-listesi > bitir

=> başla < atama > ; < ifade-listesi > bitir

=> başla < değişken > = < ifade > ; < ifade-listesi > bitir

=> başla X = < ifade > ; < ifade-listesi > bitir

=> başla X = < değişken > + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + < değişken > ; < ifade-listesi > bitir

=> başla X = Y + Z ; < ifade-listesi > bitir

=> başla X = Y + Z ; < atama > bitir

=> başla X = Y + Z ; < değişken > = < ifade > bitir

=> başla X = Y + Z ; Y = < ifade > bitir

=> başla X = Y + Z ; Y = < değişken > bitir

# Gramer ve Türetme

**<program> → başla < ifade-listesi > bitir**

**<ifade-listesi> → <atama>**

**| <atama> ; <ifade-listesi >**

**<atama> → <değişken> = <ifade>**

**<değişken> → X | Y | Z**

**<ifade> → <değişken> + <değişken>**

**| <değişken> - <değişken>**

**| <değişken>**

**<program> => başla < ifade-listesi > bitir**

**=> başla < atama > ; < ifade-listesi > bitir**

**=> başla < değişken > = <ifade> ; <ifade-listesi > bitir**

**=> başla X = <ifade> ; <ifade-listesi > bitir**

**=> başla X = <değişken> + <değişken> ; <ifade-listesi > bitir**

**=> başla X = Y + <değişken> ; <ifade-listesi > bitir**

**=> başla X = Y + Z ; <ifade-listesi > bitir**

**=> başla X = Y + Z ; <atama> bitir**

**=> başla X = Y + Z ; <değişken> = <ifade> bitir**

**=> başla X = Y + Z ; Y = <ifade> bitir**

**başla X = Y + Z ; Y = <değişken> bitir**

**=> başla X = Y + Z ; Y = Z bitir**