



BİLGİSAYAR BİLİMLERİNE GİRİŞ II -1-

DEÜ Fen Fakültesi Bilgisayar Bilimleri Bölümü
Dr. Öğr. Üyesi Resmiye Nasiboğlu

BIL1012

İlk Ders

Tanışlıq... Tekrar... Dönem Ders İçeriği.

1. Dönem sırasında bu ders bize ne kazandırdı...

- Bilgisayar kullanıcısı olarak başladık,
 - Bilgi ve algoritmalar nedir, programlama nasıl yapılır,
 - Bilgisayarın dili nedir, onunla nasıl anlaşabiliriz, onunla nasıl konuşabileceğiz diye,
- Profesyonel bilgisayarıcı olmak yolunda ilk adımı atmış olduk.
- https://www.onlinegdb.com/online_c_compiler
- <https://ideone.com/>

Bilgi

- «Enformasyon» kelimesini tez-tez duyuyoruz. Anlamı bilgi, haber, malumat, bir şeyi öğrenme, anlatma ve s.
- Bilginin her hangi bir şekilde araştırılması, incelenmesi, onun üzerinde işlem yapılması İformatik bilim dalının önemli kısmını oluşturur.
- Algoritmalarla tanıştınız; Bir amaca ulaşmak için, bir sonuç almak için basit adımlardan oluşan faaliyet kurallarıdır. Doğrusal, dallanan ve tekrarlanan algoritmalar olarak 3 tipi, ve birkaç özelliği var.
- Algoritmalar doğal dilde, grafiksel olarak ve kodlarla gösterilebilirler.
- Bilgisayarda işlem yapılarak sonuç veren algoritmalar Program adlanır.

Bilgisayar Bilimleri Nedir?

(What is Computer Science?)

- **Bilgisayar Bilimleri:**

Hesaplama(Computation) ve hesaplamanın uygulamalarını, bilgi, protokoller ve algoritmalar ışığında teorik ve pratik yaklaşımlarla ele alan bir bilim dalıdır.

- **Bilgi:**

İletim, veri, manipülasyon vb. işler için uygun formlarda veri temsili.

- **Protokol:** Bilgi alışverişindeki kurallar.

- **Algoritma:** Eylemlerin basit adımlar halinde sonlu ve kesin tanımları.

- **Hesaplama (Computation):** Bilgi işlenmesi ile ilgili genel bir terimdir.

Çoğunlukla sayısal veri işlenmesi için kullanılsa da, en dar anlamıyla [hesaplama](#) ([calculation](#)) ile, insan düşünmesine kadar uzanan olgular için kullanılan bir kavramdır.

Bilgisayar Bilimleri Nedir?

- Temel Bilgisayar bilimci neyin doğru olduğunu, hipotezin nasıl test edileceğini ve alanındaki bilgiyi nasıl geliştireceğini öğrenir.
- Bilgisayar Alanındaki Mühendis neyin kullanışlı olduğunu ve iyi anlaşılmış bir bilginin pratik bir probleme nasıl uygulanacağını öğrenir.
- Bu fark bilgisayar disiplinleri arasında da aynıdır.
- Bilgisayar bilimcisinin amacı, alanındaki var olan problemlere yeni çözüm metotları üretmek, var olan metotları geliştirmektir. Bu amacı için araç olarak yazılım kullanır.
- Bilgisayar alanındaki bir mühendis ise (Yazılım, Bilgisayar, Elektronik vb..) var olan problemin çözümü için öğrendiği metotlardan en uygun, en kolay uygulanabilir ve en verimli olacağını düşündüğünü hayata geçirmek için bir yazılım geliştirme amacındadır.
- Dolayısıyla yazılım Bilgisayar Bilimci için araç, Bilgisayar/Yazılım Mühendisi için amaçtır.

Bire Tümlleme

- **Bire Tümlleme**

- Bire tümlleme yönteminde negatif tamsayılar hesaplanırken yalnızca pozitif sayının tümlenyeni alınır. Yani 1 ise 0, 0 ise 1 yazılır.

$$0110 = 6$$

$$1001 = -6$$

- Çıkarma işlemi yapılırken sayının negatif temsili bulunur ve toplama işlemi yapılır.
- Taşan bit varsa sonuca eklenir.

$$5 - 3 = 2$$

$$0101 + 1100 = 10001$$

$$0001 + 1 = 0010 = 2$$

İkilik	Onluk
0000	0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	0

İkiye Tümlleme

■ İkiye Tümlleme

- İkiye tümlleme yönteminde negatif tamsayılar hesaplanırken önce pozitif sayının tümleyeni alınır, sonra sayıya 1 eklenir.

$$0110 = 6$$

$$1010 = -6$$

- Çıkarma işlemi yapılırken sayının negatif temsili bulunur ve toplama işlemi yapılır.
- Taşan bit dikkate alınmaz.

$$5 - 3 = 2$$

$$0101 + 1101 = 10010$$

$$0010 = 2$$

İkilik	Onluk
0000	0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Alıştırma

□ $2 - 5 = ?$ (4-bit signed representation)

1'e tümlleme

1. $0010_2 = 2$
 $1010_2 = -5$

2. $0010_2 + 1010_2$
 $= 1100_2$

3. Taşan bit yok.
 $1100_2 = -3$

2'e tümlleme

1. $0010_2 = 2$
 $1011_2 = -5$

2. $0010_2 + 1011_2$
 $= 1101_2$

3. Taşan bit yok.
 $1101_2 = -3$

Alıştırma

□ $7 - 3 = ?$ (4-bit signed representation)

1'e tümlleme

1. $0111_2 = 7$
 $1100_2 = -3$

2. $0111_2 + 1100_2$
 $= 10011_2$

3. $0011_2 + 1_2$
 $= 0100_2 = 4$

2'e tümlleme

1. $0111_2 = 7$
 $1101_2 = -3$

2. $0111_2 + 1101_2$
 $= 10100_2$

3. **Taşan bit dikkate alınmaz**
 $0100_2 = 4$

Alıştırma

- Aşağıda verilen işlemleri hem 1'e tümlleme hem de 2'ye tümlleme kullanarak 5 bitlik işaretli tamsayılarla gerçekleştirin.

	<u>1'e tümlleme</u>	<u>2'ye tümlleme</u>
1. $10 + 2 = ?$	01100_2	01100_2
2. $3 - 7 = ?$	11011_2	11100_2
3. $15 - 1 = ?$	01110_2	01110_2

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)

Polya, George (1957) '**How To Solve It**',
Princeton University Press, 2nd Edition

Algoritmaya Giriş

- **El Harezmi – Algoritma**

- Algoritma kelimesi ismini 9. yüzyıl alimi *Musa el Khowarizmi (el Harezmi)* tarafından 825 yılında yazılmış olan *Kitap al jabr w'al muqabala (Cebir ve kıyaslama kitabı)* kitabından almıştır.
- “Bir algoritma; Girdi olarak bir değer veya değer kümesi alan ve çıktı olarak bir değer veya değer kümesi üreten iyi tanımlanmış herhangi bir hesaplama prosedürü. Dolayısıyla bir algoritma, girdiyi çıktıya dönüştüren bir dizi hesaplama adımlarından oluşur. ”
- Ancak algoritma kavramı çok daha öncesine dayanır...
 - Babillerin M.Ö. 1800'lü yıllarda yazdığı düşünülen tabletlerde algoritmik işlemlerin yer aldığı görülmüştür.
 - İnanması güç olsa da bu tabletlerde çarpanlara ayırma ve kök bulma algoritmaları yer almaktadır.



Algoritmaya Giriş

- Algoritma tam olarak ne demektir?
 - Bir problemi çözmeye yarayan, sonlu, iyi tanımlanmış, sıralı işlemler kümesidir.
 - Adım adım ilerleyen bir hesaplama prosedürüdür.
- Aşağıdaki algoritma örneklerine bakalım.

*Fen Fakültesi Binasından Yemekhaneye
Gitme Algoritması*

1. BAŞLA.
2. Binanın **ana kapısından dışarı çık.**
3. **Önündeki yola** girip sağa dön ve döner kavşağa gelene kadar yürü.
4. Sol ön çaprazında gördüğün 4 katlı binaya gir.
5. 2. kata çık. *// Yemekhaneye vardın!!!*
6. BİTİR.

Sınıfta Soru Sorma Algoritması

1. BAŞLA.
2. Hoca konuşuyor mu? (*Evet/Hayır*)
3. 'Evet' ise Adım 4'e git, 'Hayır' ise Adım 5'e git.
4. Hocanın lafını bitirmesini bekle.
5. Elini kaldır ve söz iste.
6. Hoca sana söz verdi mi? (*Evet/Hayır*)
7. 'Evet' ise sorunu sor, 'Hayır' ise Adım 2'ye git.
8. BİTİR.

Algoritmaya Giriş

Bitiş: DEÜ Kaynaklar Yerleşkesi, Kuruçeşme Mahallesi,...


6 dk. (500 m)

DEÜ Kaynaklar Yerleşkesi üzerinden
Çoğunlukla düz

↑ DEÜ Kaynaklar Yerleşkesi adlı yerden kuzeybatı
yönünde ilerleyin
39 m

➔ DEÜ Kaynaklar Yerleşkesi boyunca ilerlemek için
sağa dönün
400 m

📍 Döner kavşaktan 3. çıkışa girin ve DEÜ Kaynaklar
Yerleşkesi boyunca ilerleyin



Harita verileri ©2017 Google Şartlar www.google.com/maps 100 m

Geri bildirim gönder

Algoritmaya Giriş

- Bir algoritmanın 4 (bazı kaynaklara göre 5, 1. özellik 2 ayrı özellik olarak da gösterilebiliyor) özelliği olmalıdır:
 - Valid Input / Output (Geçerli Giriş/Çıkış, Sonuç)
 - Finiteness (Sonluluk)
 - Herhangi bir giriş için, algoritma sınırlı sayıda adımdan sonra sona ermelidir.
 - Definiteness (Kesinlik, açıklık)
 - Algoritmanın tüm adımları kesin olarak tanımlanmalıdır.
 - Effectiveness (Etkinlik)
 - Algoritmanın her adımını doğru bir şekilde ve sınırlı bir süre içinde gerçekleştirmek mümkün olmalıdır. Her adımın kesin olması yeterli değildir (veya kesin olarak tanımlanmış), ancak aynı zamanda mümkün olmalıdır.

Algoritmaya Giriş

- Alıştırma:

İki sayıdan **küçük** olanı bulan algoritmayı yazınız.

Algoritma :

1. Başla
2. Kullanıcı iki sayı girsin. (x, y)
3. x, y'den **küçükse** x'i yazdır, değilse y'yi yazdır.
4. Bitir

Example:

Step Form

1. START
2. Read the first number
3. Read the second number
4. Find the sum of the two numbers
5. Print the sum of the numbers
6. STOP.

Pseudocode

1. Start
2. Read X
3. Read Y
4. $\text{Sum} \leftarrow X + Y$
5. Print Sum
6. STOP.

Example: Calculating the area of a triangle

1. START
2. Read the base
3. Read the height
4. Multiply the base by the height, and then divide by 2
5. Print the result
6. STOP.

Example:

Step Form

1. START
2. Read the base
3. Read the height
4. Multiply the base by the height, and then divide by 2
5. Print the result
6. STOP.

Pseudocode

b-base, h-height, area-A

1. START
2. Read b
3. Read h
4. $A = (b * h) / 2$
5. Print A
6. STOP.

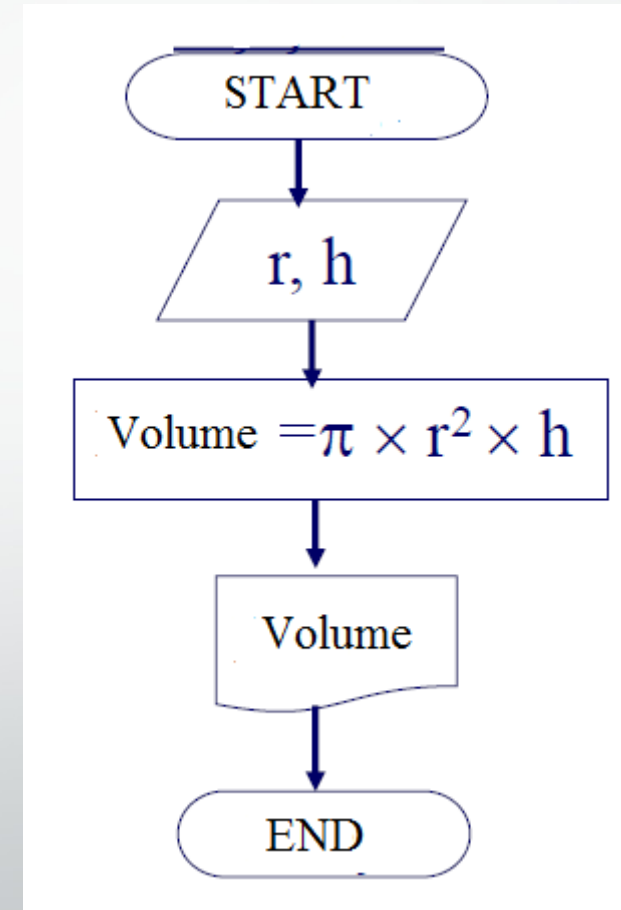
Algoritmaya Giriş

- Alıştırma:
- İki sayıdan büyük olanı bulan algoritmayı yazınız.
- Üç sayıdan büyük olanı bulan algoritmayı yazınız.
- Üç sayıdan küçük olanı bulan algoritmayı yazınız.
- Üç sayının ortancasını bulan algoritmayı yazınız.
- 5918 sayıdan büyük olanı bulan algoritmayı yazınız.
- Verilen 5 adet sayının ortalamasını bulan algoritmayı yazınız.
- Vize, final ve ödev notları verilen bir öğrencinin dönem sonu not ortalamasını bulan algoritmayı yazınız. (Vize[%20], Ödev [%40], Final [%40])
- Bir A4 kağıdı kullanarak uçak yapma algoritmasını yazınız.

Örnek: Silindirin hacmini hesaplayan algoritma yazınız ve akış diyagramı çiziniz.

Step Form

1. START
2. Read height and radius
3. Find a volume of a cylinder
4. Print the result
5. STOP.



Flowcharts – Akış Şemaları

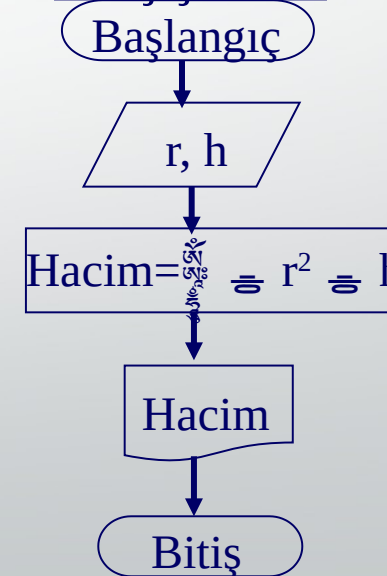
- Alıştırma:

Gerekli parametreleri kullanıcı tarafından girilen bir silindirin hacmini hesaplayıp ekrana yazdırın.

Algoritma :

1. Başla.
2. Kullanıcı yarıçapı (r) ve yüksekliği (h) girsin.
3. Silindirin hacmini bulun.
$$\text{Hacim} = \pi \cdot r^2 \cdot h$$
4. Silindirin hacmini yazdırın.
5. Bitir

Akış şeması :



Programlama Dillerine Giriş

- Dilin özellikleri
 - Alfabeti
 - Gramer, yazım kuralları (syntax)
 - Anlamı (semantics)
- Türkçe ile Japonca'yı karşılaştırınız.
- Programlama dili nedir?
 - Herhangi bir algoritmayı (çözüm tarifini) ve yapılacak işlemleri bilgisayar tarafından anlaşılıp çalıştırılmak üzere geliştirilmiş, yapay bir dildir.

Programlama Dillerine Giriş

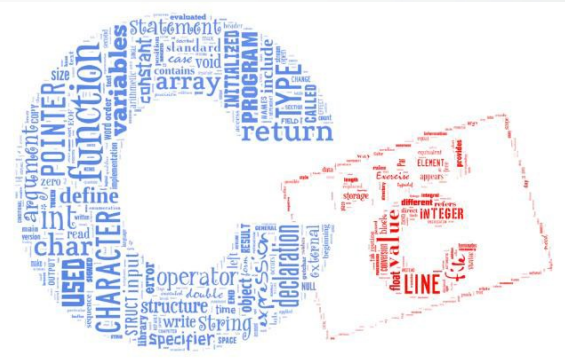
- Programlama dillerinin değerlendirme kriterleri:
 - Readability (Okunabilirlik)
 - Writability (Yazılabilirlik)
 - Reliability (Güvenilirlik)
 - Portability (Taşınabilirlik)
 - Cost (Maliyet)
 - Modelling capability (Modelleme yeteneği, becerisi)
 - Programming Environment (Programlama ortamı, editörü)

Programlama Dillerine Giriş

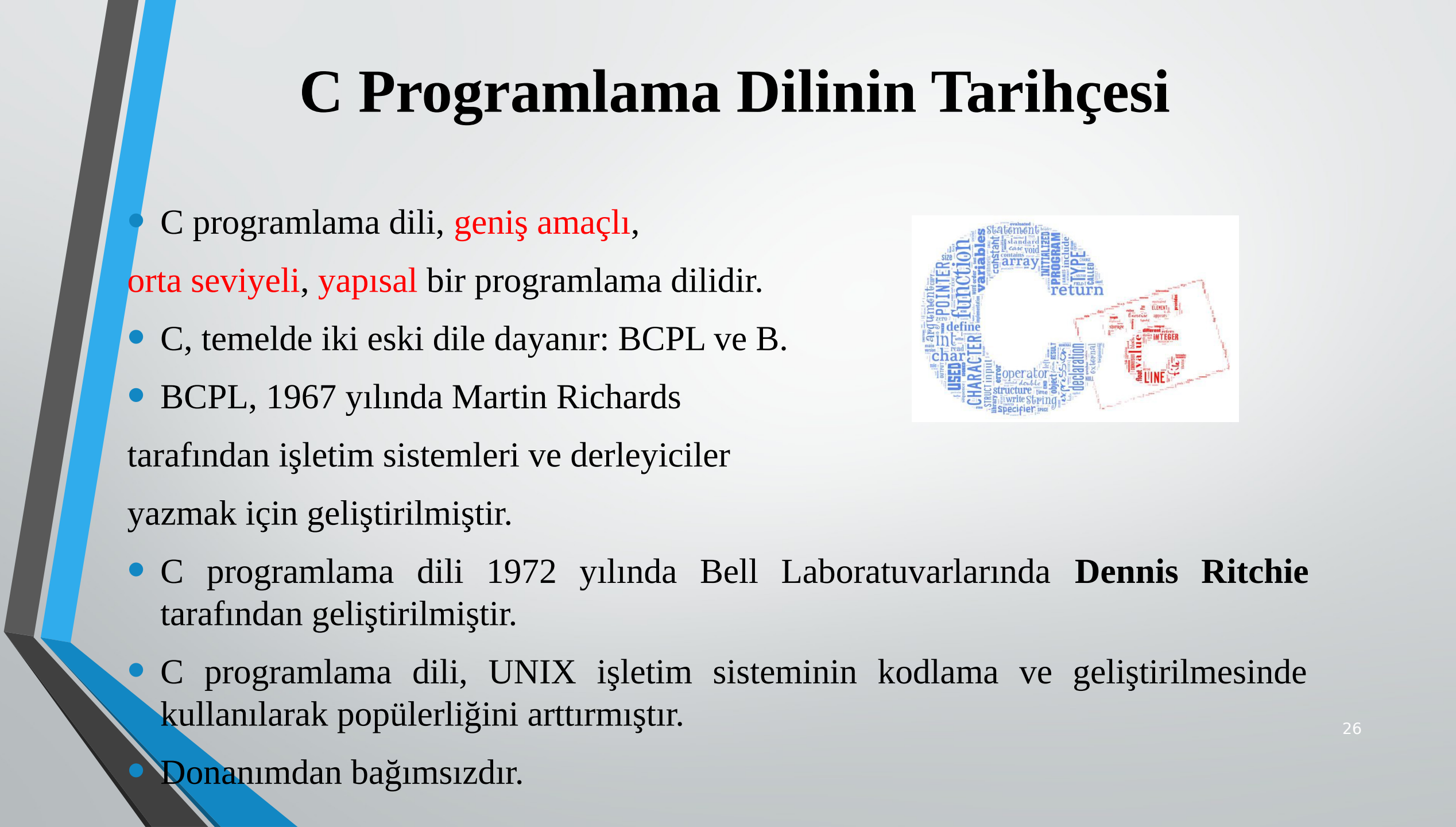
- Temel Programlama Elemanları
 - Değişkenler ve Sabitler
 - Operatörler
 - Koşul Yapıları
 - Döngü Yapıları
 - Fonksiyonlar/Prosedürler

C Programlama Dilinin Tarihçesi

- C programlama dili, **geniş amaçlı**, **orta seviyeli**, **yapısal** bir programlama dilidir.
- C, temelde iki eski dile dayanır: BCPL ve B.
- BCPL, 1967 yılında Martin Richards tarafından işletim sistemleri ve derleyiciler yazmak için geliştirilmiştir.
- C programlama dili 1972 yılında Bell Laboratuvarlarında **Dennis Ritchie** tarafından geliştirilmiştir.
- C programlama dili, UNIX işletim sisteminin kodlama ve geliştirilmesinde kullanılarak popülerliğini arttırmıştır.
- Donanımdan bağımsızdır.



- [illegible]



C Programlama Dilinin Tarihçesi

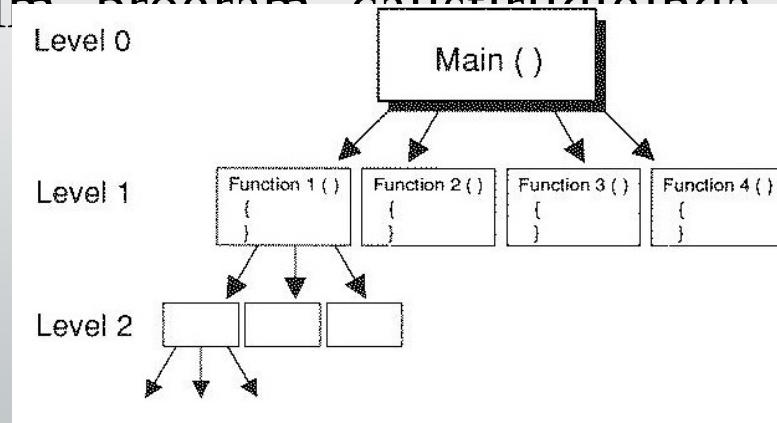
- 1970'lerin sonunda, C şu anda geleneksel C olarak bilinen haline geldi.
- Zamanla C'nin yayılması, birbirine benzer ama genellikle uyumsuz, bir çok çeşidinin ortaya çıkmasına sebep oldu.
- 1983 yılında, *American National Standards Committee*'nin bilgisayar ve bilgi işlem komitesi tarafından C'nin sistem bağımsız bir tanımı yapıldı.
- 1989 yılında bu standart onaylandı ve 1999 yılında tekrar gözden geçirildi.

C Programlama Dili Tercih Nedeni

- En **popüler** dillerden birisidir.
- Güçlü ve **esnek** bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafikler çizebilirsiniz.
- Yazılım geliştirme ortamları oldukça fazladır.
- Özel komut ve veri tipi tanımlamasına izin verir.
- **Taşınabilir** bir dildir.
- Gelişimini tamamlamış ve **standardı oluşmuş** bir dildir.
- **Yapısal** bir dildir. C kodları fonksiyon olarak adlandırılan alt programlardan oluşmuştur.

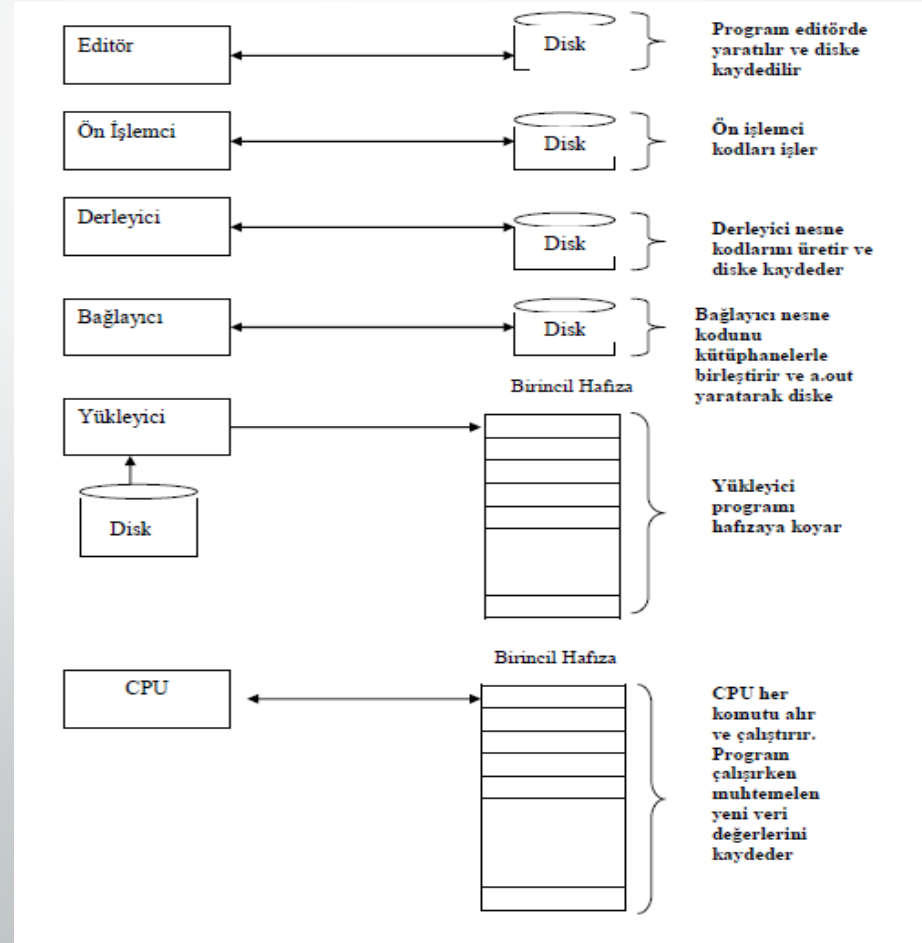
Standart C Kütüphanesi

- C programları «fonksiyon» adı verilen parçalardan yada modüllerden oluşur.
- Fonksiyonlar C «bloklarından» oluşur.
- Her fonksiyon/blok bir veya daha fazla «deyimi» içerir.
- Her bir deyim program çalıştırıldığında belirli bir eylemi yerine getirir.



C Programı Geliştirme Ortamının Temelleri

- Tipik olarak bir C programı çalışmadan önce altı safhadan geçer.
 - Yazım(Edit)
 - Önişleme(Preprocess)
 - Derleme(Compile)
 - Bağlama(Link)
 - Yükleme(Load)
 - Çalıştırma(Execute)



C Dilinin Temel Yazım Özellikleri

- Program yazımı bloklar halinde olur.
- Bloklar, { } parantezleri ile oluşturulur.
- Komutlar aynı veya alt alta satırlara yazılabilir.
- Tüm komutlar, noktalı virgül (;) ile biter.
- Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
- Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
- Programda kullanılacak olan komutların bulunduğu kütüphaneler aktifleştirilir/çağırılır.

C Program Yapısı

Ön işlemci Direktifleri
(Preprocessor Directives)

Genel Tanımlamalar
(Global Declarations)

```
main ( )
```

```
{
```

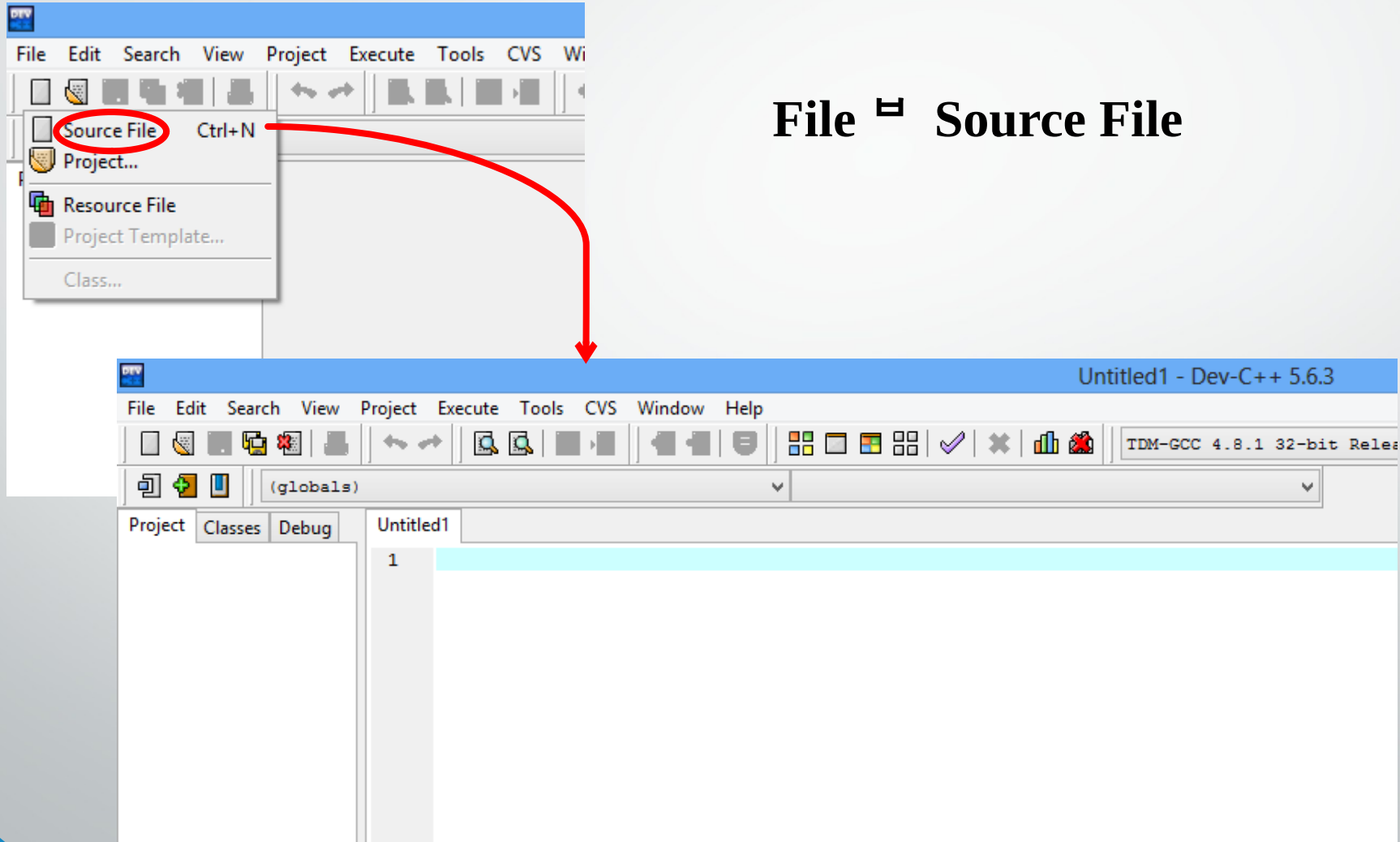
Yerel Tanımlamalar (Local Declarations)

Deyimler ve İfadeler (Statements)

```
}
```


C File Açma

File ≡ Source File



İlk Programım

```
/* C ile ilk program*/
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    printf("Merhaba Dünya!");
```

```
}
```

Ekran görüntüsü;

Merhaba Dünya!

/* ve */ arasına yorum yazılır, derleyici görmezden gelir.

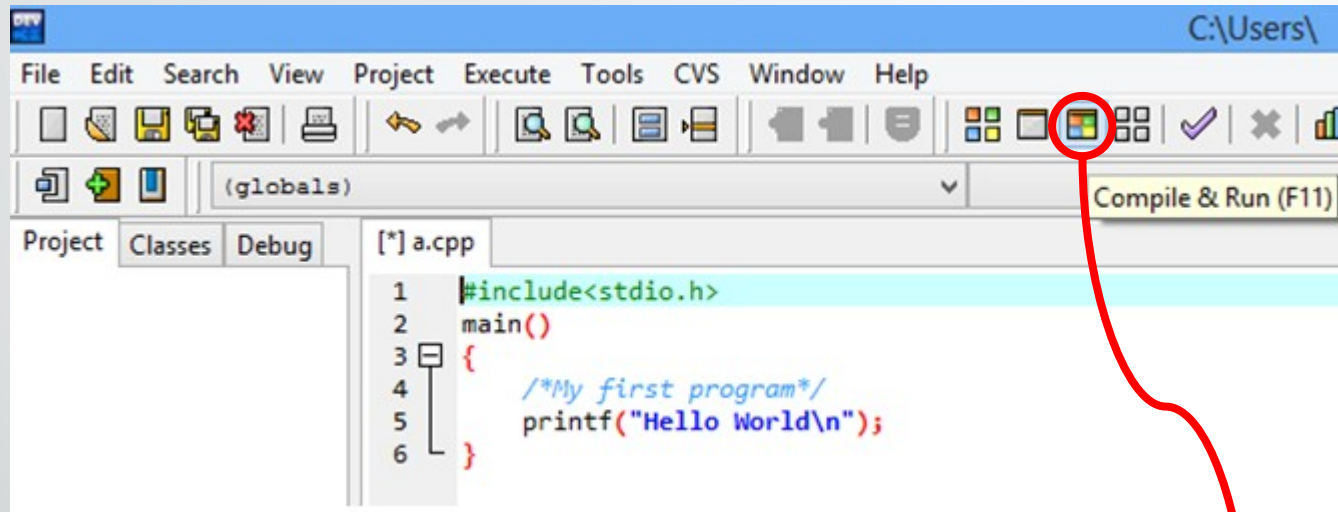
#include C önışlemcisine bir emir göndermektedir. Bu satır, önışlemciye standart giriş/çıkış öncü dosyası(stdio.h) içeriğinin programa eklenmesini söyler.

void main() her C programının bir parçasıdır. C programları bir veya birden fazla fonksiyon içerebilir ancak bunlardan biri mutlaka main olmalıdır. C'de her program main fonksiyonunu çalıştırarak başlar.

Küme parantezi, { , her fonksiyonun gövdesinin başına yazılır. }, küme parantezi ise sonuna yazılmalıdır. Bu iki parantez arasında kalan program parçacığına blok denir.

printf konsola yazdırma işlemini gerçekleştirir.

Compile \equiv Run (F11)



Hello World

Process exited after 0.008779 seconds with return value 0
Press any key to continue . . . _

Yorum Satırı

- **Tekli ve Çoklu Yorum Satırı**
 - `//` Tek satırda yorum yapılacağı zaman kullanılır.
 - `/*` Çoklu yorum satırı `*` bir satırla açıklama yapamayacağımız zaman `*` kullandığımız bir yöntemdir.
`*/`

C Programlama Dili Elemanları

- Anahtar Sözcükler
- Veri Türleri
- Değişkenler
- Sabitler
- Operatörler

C Anahtar Kelimeleri

Anahtar Kelimeler (Keywords)			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C Veri Türleri

- **Veri tipi (data type)** program içinde kullanılacak değişken, sabit, fonksiyon isimleri gibi tanımlayıcıların tipini, yani bellekte ayrılacak bölgenin büyüklüğünü, belirlemek için kullanılır.
- Bir programcı, bir programlama dilinde ilk olarak öğrenmesi gereken, o dile ait veri tipleridir. Çünkü bu, programcının kullanacağı değişkenlerin ve sabitlerin sınırlarını belirler.

C Veri Türleri

- C programlama dilinde 5 tane temel veri tipi bulunmaktadır.
 - **char**: karakter veriler
 - **int**: tamsayı veriler
 - **float**: tek duyarlıklı kayan noktalı sayılar
 - **double**: Çift duyarlıklı kayan noktalı sayılar
 - **void**: Değer içermeyen verilerdir.

C Değişkenleri

- Değişken, program içinde kullanılan değerlere bellek üzerinde açılan alanlardır. Bu alanlar bir **değişken ismi** ile anılır.
- Değişken isimlendirilmeleri, tanımlayıcı kurallarına uygun biçimde yapılmalıdır.
- C’de tüm değişkenler kullanılmadan önce programa bildirilmelidir.
- Bu bildirim esnasında, değişkenin veri türü belirlenir.
- **Örnek:**

veri_türü değişken_adı;

int sayac;

C Sabitleri

- Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır.
- Ancak, veri tipinin önüne **const** anahtar sözcüğü konmalıdır.
- Sabit içerikleri **program boyunca değiştirilemez**. Yalnızca kullanılabilir.
- Genellikle, sabit olarak bildirilen değişken isimleri **büyük harflerle**, diğer değişken isimlerinin ise küçük harflerle yazılması (gösterilmesi) C programcıları tarafından **geleneksel** hale gelmiştir.

Örnek: C Sabitleri

- Örnekler:
 - `const float PI=3.142857;`
 - `const double NOT=12345.8596235489;`
 - `const int EOF=-1;`
 - `const char[]="devam etmek için bir tuşa basın...";`

Tam Sayılar - Integer

- Tam sayıları ifade eder
 - Hem negatif hem pozitif tam sayılar
- C de tam sayıların (integer) ifade tarzı: int
- Örnek:

```
int toplam;    /* işaretli integer */
```

```
toplam = 100; /* pozitif olabilir */
```

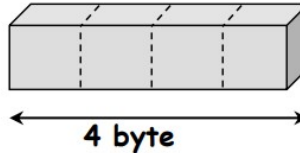
```
toplam = -20; /* negatif olabilir */
```

```
int toplam = 32000; /* kodlama sırasında */  
                  /* ilk değer verilebilir */
```

Tam Sayılar – Integer (Devam)

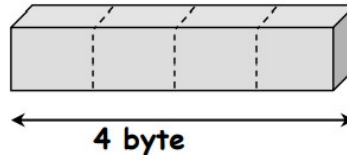
- Integer niteleyicileri: long, short, veya unsigned
- Integer değişkenlerin niteleyicilerine göre büyüklükleri değişir.
- Varsayılan integer büyüklüğü makine/işletim sistemine ba.dır

int



-2.147.483.648 den 2.147.483.647 e kadar (toplam 4.294.967.296 adet sayı)

**unsigned
int**



**0 dan 4,294,967,295 e kadar
(toplam 4,294,967,296 adet sayı)**

Virgüllü Sayılar - float

- Gerçek sayıları ifade eder (virgüllü kısmıyla)
 - Pozitif ve negatif olabilir
- C de virgüllü sayıların ifade tarzı: float
- Örnek:

```
float f;
```

```
f = 0.12;    /* pozitif olabilir */
```

```
f = -245.56; /* negatif olabilir */
```

```
float f = 4.567; /* kodlama sırasında      */  
                /* ilk değer verilebilir */
```

Daha Uzun ve Çok Hassas Virgüllü Sayılar- double

- Standart "double precision floating point" (gerçek) sayılardır.
 - float gibi, fakat çok daha büyük ve hassastır.
- C deki ifade tarzı: double
- Örnek:

double	8 bayt	2.3E-308 , 1.7E+308
long double	10 bayt	3.4E-4932 , 1.1E+4932

Karakter - char

- Bir tek karakteri ifade eder
 - Karakterler
 - Alfabedeki büyük ve küçük harfler
 - 0 dan 9 a kadarki 10 numara
 - Özel semboller örneğin
 - +#@½%&\$.*?!£‘=-:/*^(){}[]~;, <>
- Karakterler tırnak işareti arasında kullanılır
 - örneğin. 'A'
- C deki kullanım tarzı: char

```
char c;
```

```
c = 'A'; /* A Harfi */
```

```
c = '9'; /* 9 rakamı*/
```

```
char c = 'c'; /* ilk değer verme */
```

Karakter (devam)

- Aslında karakterler 1 byte lık doğal sayıları ifade eder
 - char tipi değişkenler hafızada 1 byte yer tutar
- Karakterlerin (char değişkenleri) ASCII tablosundaki değerleri...
 - 'A' nın ASCII değeri 65
 - 'B' nın ASCII değeri 66
 - '0' ın ASCII değeri 48
 - '1' in ASCII değeri 49
 - <http://www.asciitable.com/>

ASCII Tablosu

Decimal Hex Char			Decimal Hex Char			Decimal Hex Char			Decimal Hex Char		
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[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	[NEGATIVE 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	[ENG OF 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	[SUBSTITUTE]	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	[DEL]

Veri Tipleri ve Özellikleri

Veri Tipi	Açıklama	Bellekte işgal ettiği boyut (bayt)	Alt sınır	Üst sınır
char	Tek bir karakter veya küçük tamsayı için	1	-128	127
unsigned char			0	255
short int	Kısa tamsayı için	2	-32,768	32,767
unsigned short int			0	65,535
int	Tamsayı için	4	-2,147,483,648	2,147,483,647
unsigned int			0	4,294,967,295
long int	Uzun tamsayı için	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long int			0	18,446,744,073,709,551,615
float	Tek duyarlı gerçel sayı için (7 basamak)	4	-3.4e +/- 38	+3.4e +/- 38
double	Çift duyarlı gerçel sayı için (15 basamak)	8	-1.7e +/- 308	+1.7e +/- 308

Ön İşlemci Direktifleri

- Başlık dosyaları, derleyicinin kütüphane fonksiyonu çağrılarının doğru yapılıp yapılmadığını anlamasında yardımcı olan bilgiler içerir.
- ANSI C'deki standart başlık dosyaları aşağıdaki gibidir:

- ❖ assert.h
- ❖ ctype.h
- ❖ errno.h
- ❖ float.h
- ❖ limits.h

- ❖ locale.h
- ❖ math.h
- ❖ setjmp.h
- ❖ signal.h
- ❖ stdarg.h

- ❖ stddef.h
- ❖ stdio.h
- ❖ stdlib.h
- ❖ string.h
- ❖ time.h

Escape Sequence – Kaçış Dizisi

- Ters slash (\) ve bir karakterden oluşur. Derleyiciye sonraki karakterin normal olarak algılanması işaretini verir.
- Sık kullanılanlar
 - \n sonraki satıra geç
 - \t sonraki sekmeye geç
 - \r satır başına alır
 - \\ ters slash karakteri –
 - \' tek tırnak
 - \" çift tırnak

Format Belirteçleri

Belirleyici	Biçim
%d, %i	Tamsayı (Decimal)
%u	İşaretsiz Tamsayı (Unsigned)
%f	Kayan Noktalı Sayı (Float)
%c	Karakter (Char)
%o	8 Tabanında Sayı (Octal)
%x, %X	16 Tabanında Sayı(Hexadecimal)
%e	Üssel Gösterim (Exponential)
%s	Karakter Dizisi (String)
%l, %h	Long ve short ön eki

main() Fonksiyonu

- Programın özel bir fonksiyonudur; ana program anlamındadır.
- C diliyle yazılmış bir program yürütülmeye başlandığında ilk bu fonksiyon çağrılır.
- Programın yürütülmesi bu fonksiyondan başlar.

Input/Output Fonksiyonları

- I/O fonksiyonları standart input/output C Kütüphanesinde tanımlanmış
 - **stdio.h**
- Klavye Input
 - **scanf** -- Genel Formatlanmış input
 - **getchar** -- tek bir karakter okur
- Monitör (Ekran) Output
 - **printf** -- Genel Formatlanmış output
 - **putchar** -- tek bir char (karakter) yazar

printf Fonksiyonu

- Ekrana veriyi biçimlendirerek yazabilen bir fonksiyondur.

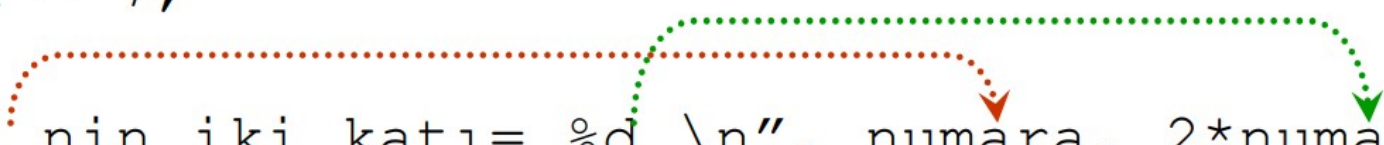
printf("biçim ifadesi", değişkenler);

- Çift tırnak arasında yer alan ‘biçim ifadesi’ genel olarak üç kısımdan oluşur.
 - Açıklama kısmı
 - Biçim kısmı
 - Kontrol/çıkış Kısmı

Örnek

```
int numara = 7;
```

```
printf("%d nin iki katı= %d \n", numara, 2*numara);
```



Biçim kısmı

İfade kısmı

printf Fonksiyonu

- **Açıklama:** Çift tırnaklar arasında verilip ekrana doğrudan yazılır.

printf("Ankara");

- **Biçim:** % sembolüyle başlayan ve çıkış biçiminin belirlendiği kısımdır.

printf("Sonuc: %d ", x);

- **.precision:** maksimum kaç karakterde gösterileceğini belirtir.

printf("Sonuc: %.2lf ", y);

printf Fonksiyon Örnekleri

```
double fp = 251.7366;  
int i = 25;  
printf("Reel sayi: %.2lf \n", fp);  
printf("Saga yaslanilmis integer: %10d \n", i);
```

Çıktı:

```
Reel sayi: 251.74  
Saga yaslanilmis integer :           25
```

printf Örnekler

```
printf("%.5f\n", 300.0123456789);  
printf("%.14lf\n", 300.01234567890123456789);
```

300.01235

300.01234567890123

scanf Fonksiyonu

- Klavyeden belirtilen değişkene veri aktarılmasını sağlar.

scanf(" biçim ifadesi ", &değişkenler listesi);

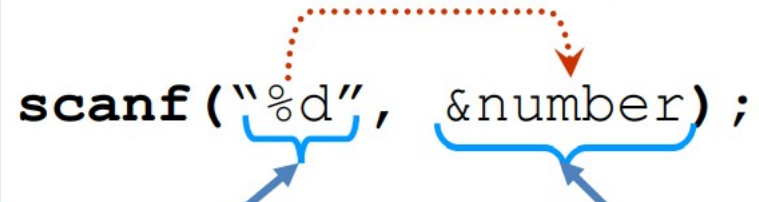
- Buradaki "biçim ifadesi" veri girişinin hangi biçimde olacağını; "değişkenler (adres) listesi" de verilerin aktarılacağı değişkenleri belirtir.

Örnek

```
int number;
```

```
printf("Bir integer girin: ");
```

```
scanf("%d", &number);
```



Biçim kısmı

Değişken adresi

Örnek: Girilen iki sayıyı ekrana yazdıran program tasarlayınız.

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int a,b;
```

```
    printf("iki sayi giriniz");
```

```
    scanf("%d %d",&a,&b);
```

```
    printf("girilen sayilar = %d, %d", a, b );
```

```
}
```

```
iki sayi giriniz78
90
girilen sayilar=78,90_
```

Alıştırmalar

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    int x;
```

```
    printf ("Bir tamsayi girin.\n");
```

```
    scanf ("%d", &x);
```

```
    printf ("Onluk tabanda = \t%d\n", x);
```

```
    printf ("Sekizlik tabanda = \t%o\n", x);
```

```
    printf ("Onaltilik tabanda = \t%X\n", x);
```

```
}
```

```
Bir tamsayi girin.  
29  
Onluk tabanda =      29  
Sekizlik tabanda =   35  
Onaltilik tabanda =  1D
```

Alıştırmalar

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    float x;
```

```
    printf ("Ondalıkli sayı girin(nokta ile).\n");
```

```
    scanf ("%f ", &x);
```

```
    printf ("Girilen Sayi = %f \n", x);
```

```
}
```

```
Ondalikli sayi girin(nokta ile).  
12.32  
Girilen Sayi = 12.320000
```

C Operatörleri

- Operatörler, değişkenler veya sabitler üzerinde matematiksel ve karşılaştırma işlemlerini yapan simgelerdir. Yani bir operatör bir veya daha fazla değişken üzerinde işlem yapan semboldür.
- C programlama dilinde 4 tip operatör bulunmaktadır.
 - Aritmetik Operatörler
 - Atama Operatörleri
 - Karşılaştırma Operatörleri
 - Mantıksal Operatörler

C Operatörleri (Aritmetik Operatörler)

Operatör	Açıklama	Örnek	Anlamı
+	toplama	$x + y$	x ve y nin toplamı
-	çıkarma	$x - y$	x ve y nin farkı
*	carpma	$x * y$	x ve y nin çarpımı
/	bölme	x / y	x ve y nin oranı
%	mod alma	$x \% y$	x / y den kalan sayı

C Operatörleri (Atama Operatörleri)

Operatör	Açıklama	Örnek	Anlamı
=	atama	x = 7;	x = 7;
+=	ekleyerek atama	x += 3	x = x + 3
-=	eksilterek atama	x -= 5	x = x - 5
*=	çarparak atama	x *= 4	x = x * 4
/=	bölerek atama	x /= 2	x = x / 2
%=	bölüp, kalanını atama	x %= 9	x = x % 9
++	bir arttırma	x++ veya ++x	x = x + 1
--	bir azaltma	x-- veya --x	x = x - 1

C Operatörleri (Atama Operatörleri)

Örnek	Anlamı
<code>x = y++;</code>	y'nin değeri önce x'e aktarılır sonra bir arttırılır. <code>x = y;</code> <code>y = y + 1;</code>
<code>x = ++y;</code>	y'nin değeri önce bir arttırılır sonra x'e aktarılır . <code>y = y + 1;</code> <code>x = y;</code>
<code>x = y--;</code>	y'nin değeri önce x'e aktarılır sonra bir azaltılır. <code>x = y;</code> <code>y = y - 1;</code>
<code>x = --y;</code>	y'nin değeri önce bir azaltılır sonra x'e aktarılır . <code>y = y - 1;</code> <code>x = y;</code>

Operatörler

• K

Operatör	Açıklama	Örnek	Anlamı
>	büyüktür	$x > y$	x, y'den büyük mü?
<	küçüktür	$x < y$	x, y'den küçük mü?
==	eşittir	$x == y$	x, y'ye eşit mi?
>=	büyük-eşittir	$x >= y$	x, y'den büyük veya eşit mi?
<=	küçük-eşittir	$x <= y$	x, y'den küçük veya eşit mi?
!=	eşit değil	$x != y$	x, y'den farklı mı?

Operatörler

•]

Operatör	Açıklama	Örnek	Anlamı
&&	mantıksal VE	$x > 5 \ \&\& \ x < y$	x, 5'den büyük VE x, y'den küçük mü?
	mantıksal VEYA	$x > 5 \ \ x < y$	x, 5'den büyük VEYA x, y'den küçük mü?
!	mantıksal DEĞİL	$!(x > 5)$	x, 5'den büyük değilse (x, 5'den küçük VEYA 5'e eşitse)

Örnek 1.

```
#include <stdio.h>
```

```
main() {
```

```
    int a = 20, b = 10, c = 15, d = 5, e;
```

```
    e = (a + b) * c / d;    // ( 30 * 15 ) / 5
```

```
    printf("değer: %d\n", e );
```

```
    e = ((a + b) * c) / d;  // (30 * 15) / 5
```

```
    e = (a + b) * (c / d);  // (30) * (15/5)
```

```
    printf(" deđer: %d\n", e);
```

```
    e = a + (b * c) / d;    // 20 + (150/5)
```

```
    printf(" deđer: %d\n" , e );
```

```
}
```

Örnek 2.

```
#include<stdio.h>

main()
{
    int m = 7, j = 7, k, z;
    k = m++;          /* k = m, m = m + 1 */
    z = ++j;          /* j = j + 1, z = j */
    printf(" %d %d %d %d", m, j, k, z);
    getch();
}
```

8 8 7 8

Örnek 3.

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    int x, y;
```

```
    printf( "İki Sayı Giriniz: " );
```

```
    scanf( "%d%d", &x, &y );
```

```
    printf( "Toplam: %d\n", x + y );
```

```
    printf( "Carpım: %d\n", x * y );
```

```
    printf( "Fark: %d\n", x - y );
```

```
    printf( "Bolum: %f\n", (float)x / y );
```

```
    printf( "Mod: %d\n", x % y );
```

```
}
```

```
İki Sayı Giriniz: 100 15
Toplam: 115
Carpım: 1500
Fark: 85
Bolum: 6
Mod: 10
```


Örnek 4.

```
#include <stdio.h>
```

```
main(){
```

```
    printf( "1 2 3 4\n\n" );  /* bir printf ifadesi ile */
```

```
    printf( "%d %d %d %d\n\n", 1, 2, 3, 4 );  /* bir printf ifadesi + 4 format belirteci ile */
```

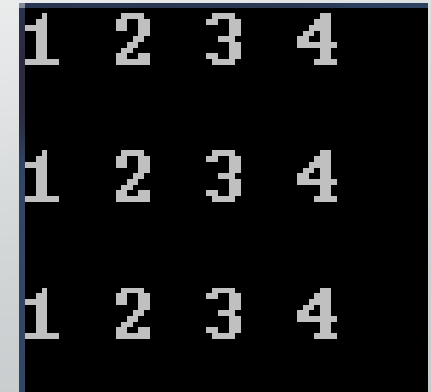
```
    printf( "1 " );          /* 4 printf ifadesi ile */
```

```
    printf( "2 " );
```

```
    printf( "3 " );
```

```
    printf( "4\n" );
```

```
}
```



1	2	3	4
1	2	3	4
1	2	3	4

Örnek5. Yarıçapı 5 olan çemberin çapını, çevresini, alanını hesaplamak için program

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    int radius;
```

```
    printf( "Yaricapı giriniz: " );
```

```
    scanf( "%d", &radius );
```

```
    printf( "\nCap: %d\n", 2 * radius );
```

```
    printf( "Cevre: %f\n", 2 * 3.14159 * radius );
```

```
    printf( "Alan: %f\n", 3.14159 * radius * radius );
```

```
}
```

```
Yaricapı giriniz: 5
```

```
Cap: 10
```

```
Cevre: 31.415900
```

```
Alan: 78.539750
```

Örnek 6(1).

- Sadece bu bölümde öğrendiğiniz programlama tekniklerini kullanarak 0'dan 10'a kadar olan sayıları karelerini hesaplayıp, sonuçları ekrana aşağıda görüldüğü biçimde yazdıran bir program yazınız.

Sayı	Karesi	Kubu
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int count = 0;
```

```
    printf( "\nSayi\tKaresi\tKubu\n" );
```

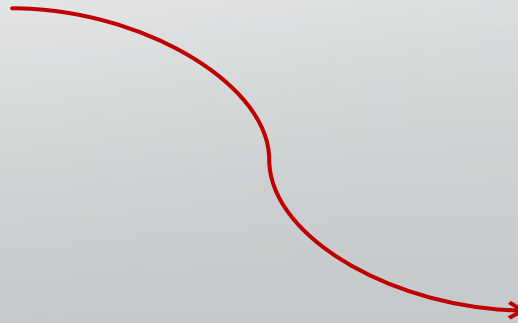
```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
    count = count + 1;
```

```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
    count = count + 1;
```

```
    printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

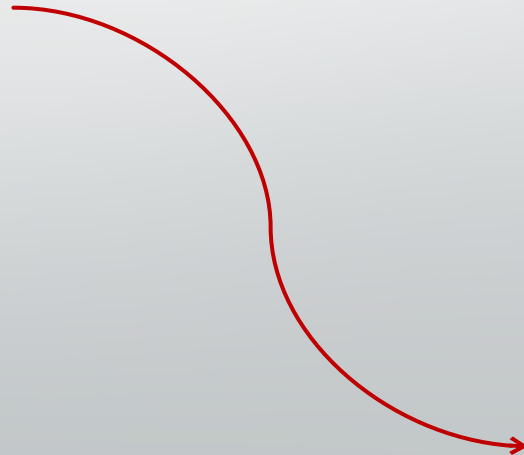


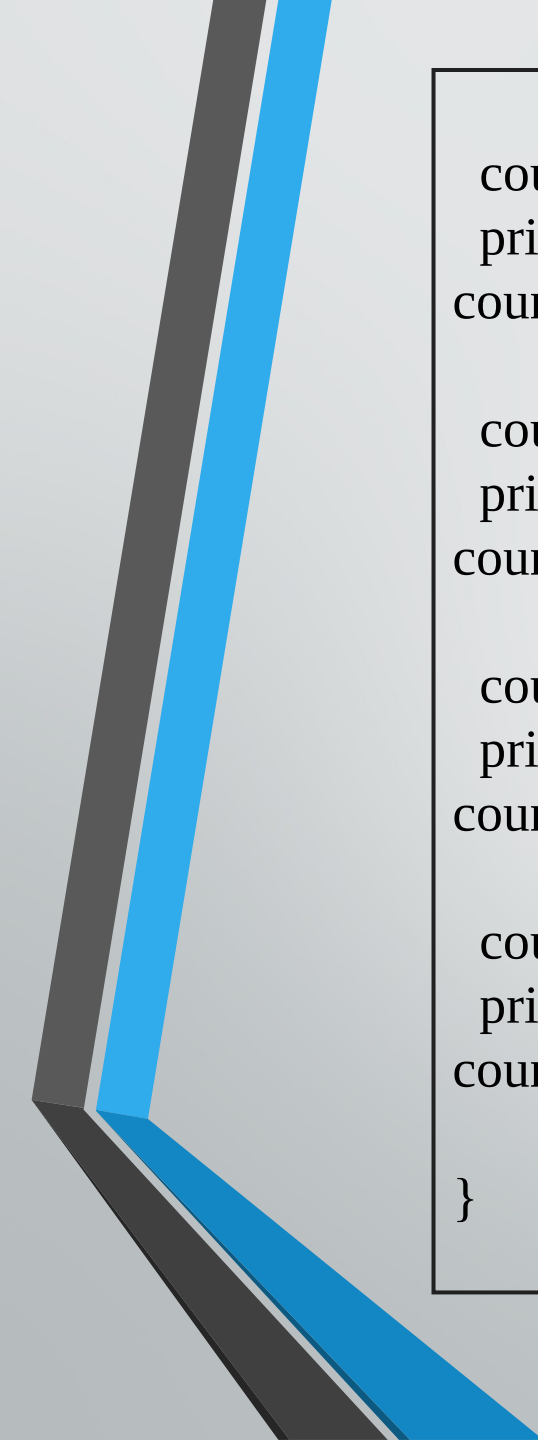
```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count * count );
```





```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count *  
count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count *  
count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count *  
count );
```

```
count = count + 1;  
printf( "%d\t%d\t%d\n", count, count * count, count * count *  
count );
```

```
}
```