

Düzce Üniversitesi
MÜHENDİSLİK FAKÜLTESİ

EEM285 C PROGRAMLAMA DERS NOTLARI

PROF. Dr. ALİ ÖZTÜRK

ÖĞRETİM ÜYESİ

Düzce, 2020

Ders Programı

- 1) Algoritma ve akış şemaları , Program Geliştirme ve Programlama dilleri,
- 2) Algoritma ve Programlama, C diline giriş, Temel girdi ve çıktı işlemleri, BloodShed Dev-C++ Ortamı, C dili ile program yazma,
- 3) C program yapısı, başlık dosyaları ve değişken tanımlama, değişken tipleri, Aritmetik Operatör ve İfadeleri, Ana kod bloğu içinde girdi-çıkı işlemleri, Kod yazma, Sayı operatörler (and, or, xor ...), Örnek sorular
- 4) Matematik fonksiyonları, karmaşık sayılar
- 5) Koşul ifadeleri, iç içe geçmiş ifadeler, if, switch case,
- 6) Döngü kavramı, for, while, do while döngüleri, Örnek çözümler
- 7) Fonksiyonlar, Örnek çözümler
- 8) Diziler (array), dizi işlemleri, Örnek çözümler
- 9) Çok boyutlu Diziler (array), Çok boyutlu dizi işlemleri, Örnek çözümler
- 10) Katarlar (String), Katarlarda printf() ve scanf() kullanımı, Örnek çözümler
- 11) Rastgele Sayı üretme
- 12) Dosya Yönetimi
- 13) Yapı ve Birlikler

KAYNAKLAR:

(Yararlanılan Temel Kaynaklar/Ders Notları:

Öğretim üyesinin ders notları, internet ortamında benzer kaynaklar.

Kitaplar/Books:

Programlama Arayüzleri (IDE):

Kullanılacak programlama ara yüzleri:

1- Dev-C++ : (Bunu kullanacağız)

[https://netix.dl.sourceforge.net/project/orwelldevcpp/Setup Releases/Dev-Cpp 5.11 TDM-GCC 4.9.2 Setup.exe](https://netix.dl.sourceforge.net/project/orwelldevcpp/Setup%20Releases/Dev-Cpp%205.11%20TDM-GCC%204.9.2%20Setup.exe)

2- CodeBlocks :

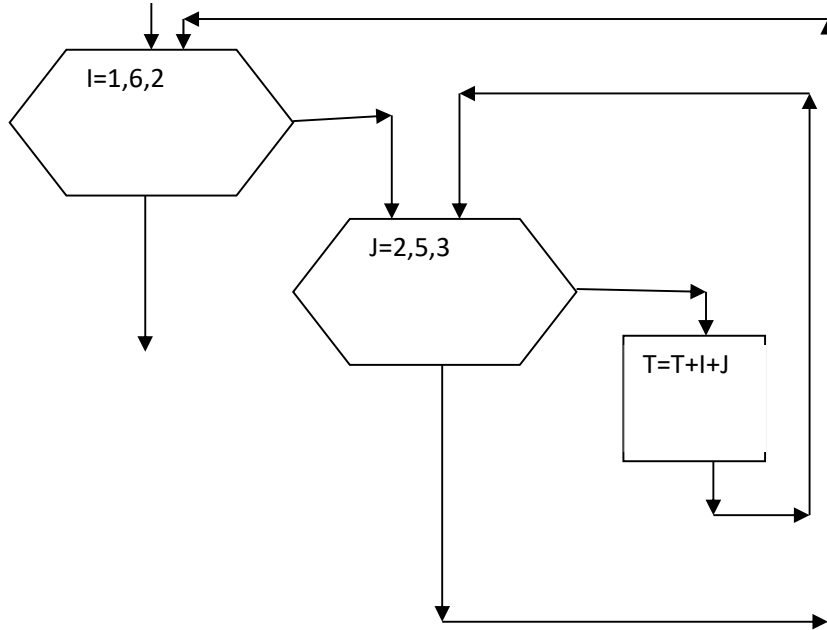
[http://www.codeblocks.org/downloads/26#windows,](http://www.codeblocks.org/downloads/26#windows)

Bu sayfadan [codeblocks-20.03mingw-32bit-setup.exe](#) indirip yükleyebilirsiniz.

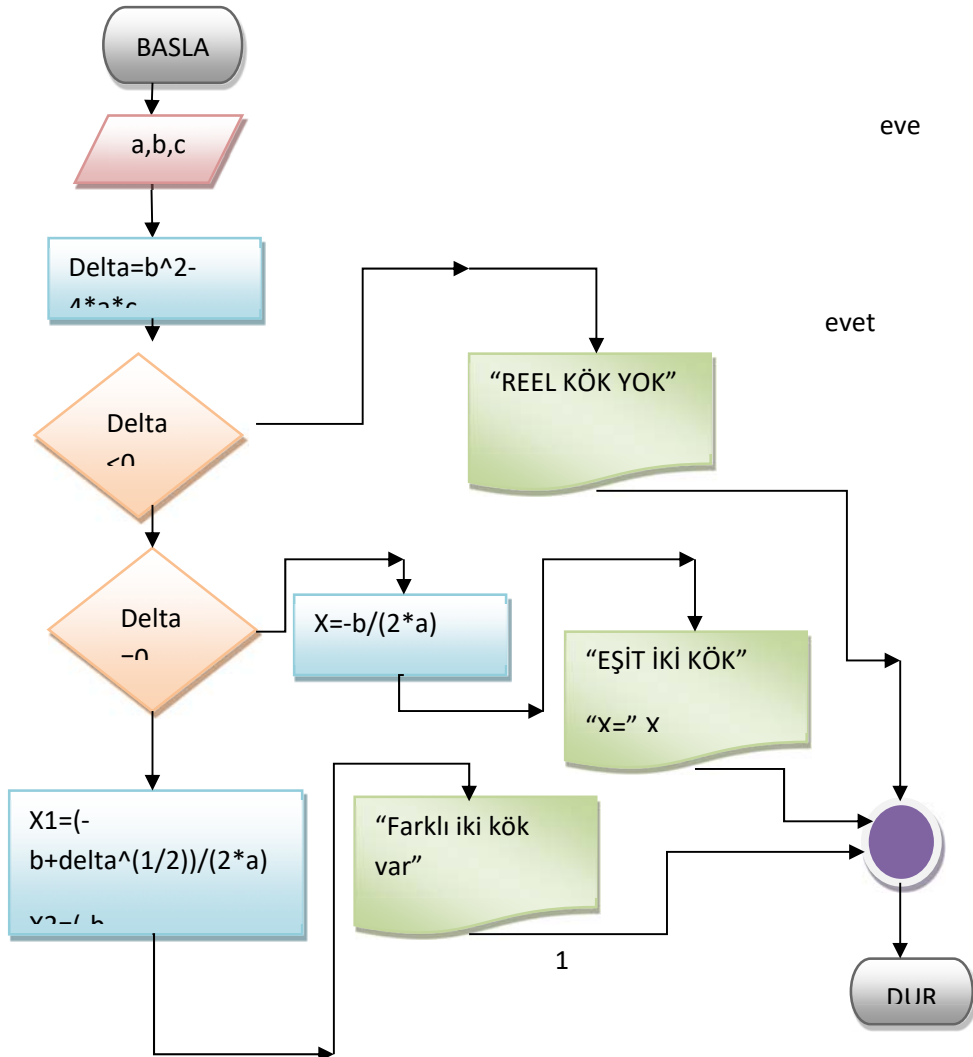
Her ikisi de kullanılabilir, biz Dev-C++ ara yüzü ile **C kodlarını** yazıp çalıştıracğız.

C PROGRAMLAMA YAZ OKULU DERS NOTU

her bir adımdaki i,j ve T değerlerini bulunuz.



ikinci Dereceden denklem katsayıları a,b,c girildiğinde denklem köklerini bulan programın akış şemasını çiziniz.



TEMEL KAVRAMLAR

1.) Bilgisayar nedir? Programlama dilleri nedir? Temel bilgisayar tanımları, programlama tanımları nelerdir?

Çok çok basit düşünürsek **bilgisayar aslında sadece bir makinedir**. Bilgisayar **üç ana işlem yapar**. Girilen veriyi alır (**Input**), bu veriyi işler (**Processing**) ve sora bu işlediği verileri bir sonuç olarak çıkarır, gösterir (**Output**). Ama bilgisayar hızlıdır, yorulmaz. Bir insan matematiksel işlemi belli bir sürede gerçekleştirirken bilgisayar bu işlem öğretildikten sonra onu, insandan çok daha hızlı gerçekleştirir. Bilgisayarı bilgisayar yapan da budur. Bilgisayar sadece donanımsal olarak bir işe yaramaz. Çünkü yukarıda belirttiğimiz “öğretilme” olgusunu bir şeyin gerçekleştirmesi gerekir ki donanım bir iş yapsın. İşte bu noktada yazılım devreye girer, programcı, yazılımcı devreye girer. Bilgisayar **donanımına ne yapacağını söyleyen komutlar dizisini yazmak oluşturmak programlamadır**.

Bilgisayarın **anladığı tek dil** vardır o da **makine dilidir**. O da 16'lık (Hexadecimal) sistemden oluşan programlama dilidir. Bu dil çok zordur ve bu dili kullanabilmeniz için bilgisayar özelliklerini tamamıyla bilmeniz gerekir. Yazıların devamında göreceğimiz bir yazdırma yani **printf(“”); kodu bilgisayar dilinde “1D BB C2 BC D5 FF C2 F7....”** gibi çok karışık bir ifadedir. **Makine dili programlama dilleri içerisinde en alt seviyededir**.

Makine dilinden sonra “Assembler Dili” gelir. Makine dilini daha kolay anlamak ve kullanmak için geliştirilen bu dil makine diline göre kolaydır. Ama yine de C dile ile karşılaştırılmaz çünkü Assembler dilinde yine donanımsal bilgiye ihtiyaç vardır.

- **Çok yüksek seviyeli diller** : VisualBasic, VB.NET, Acces , Foxpro ...
- **Yüksek seviyeli diller**: Pascal ,Basic ,Fortran...
- **Orta seviyeli diller**: C ,C++, C# , Java ,ADA...
- **Düşük seviyeli diller**: Assembly...
- **Makina dilleri**: Bilgisayarın çalışma dilleri 1 ve 0'lardan oluşur...

***Bilgisayar, Programlama ve C İle ilgili genel tanımlar:**

- **Program nedir?** : Kendi içerisinde bir bütün olan ve belirli bir işi/görevi yerine getiren algoritmik bir ifadedir.
- **Yazılım nedir?** : Yazılım genel olarak bir **işin** program **kodları üretilerek yapılmasıdır**; tanımı, donanım dışında kalan ve programcının kodlama yaparak istenilen bir işi veya görevin yerine getirilmesi için oluşturduğu program/kod ve veri kümesidir. Yazılım bir veya birden çok programın bir araya gelmesinden oluşan bir program kümesidir.
- **Program Kodu nedir?** : Bir işin yapılması için bir algoritmik ifadeyi gösteren programın herhangi bir programlama diliyle, o dilin özellikleriyle elde edilmiş program parçasıdır. Program kodu bir satırdan oluşabileceği gibi programın tamamını da kapsayabilir.
- **Değişken nedir?** : **Verilerin tutulduğu bellek gözlerine verilen simgesel isimlerdir**; dolayısıyla program içerisinde kullanılacak veriler değişkenler üzerinde saklanırlar/ tutulurlar.
- **Diziler nedir?** : **Aynı türden verilerin tek bir isim altında bellekte tutulması için kullanılan bir yöntemdir**; dizi adı da bir değişkendir.
- **Operatör nedir?** : **Değişkenler, veriler üzerinde işlem yapma özelliği olan simgelerdir**. Örneğin toplama operatörü iki değişkenin veya verinin toplanmasını sağlar. Programlama dillerinde operatörler önceden tanımlanmışlardır.
- **Donanım nedir?** : Donanım genel olarak, **elektronik elemanlardan** oluşan ve bilgisayar sistemini oluşturan işlemci,disk, ana kart, bellek vs. gibi birimlere verilen adlandırmadır.
- **Bellek nedir?** : Bellek programına ait kodların, üzerinde işlem yapılacak verilerin ve üretilen sonuçların üzerinde saklandığı **donanım parçasıdır**.
- **İşlemci nedir?** : Bilgisayar veya benzeri bir sistem içerisinde aritmetik, mantık ve karşılaştırma **işlemlerinin yapıldığı**; ve bu işlemlerin anlamlı sıralarla art arda kullanılmasıyla daha karmaşık işlemlerin kotarıldığı **programlanabilen bir birimdir**.
- **Makine Kodu nedir?** : İşlemcinin komut kümesindeki **komutlarla yazılmış bir program parçasıdır**. İşlemci, makine koduyla yazılmış programları hiçbir derleyici veya dönüştürücüye ihtiyaç duymadan çalıştırabilir.

- **Assembly Dili nedir?** : Bu dil, makine kodu düzeyinde olan komutlara **simgesel isimler verilerek oluşturulmuştur**. Çünkü makine kodu ile program yazılması güçtür.
- **İşletim Sistemi nedir?** : İşletim sistemi, bilgisayar **donanımı ile kullanıcı ve programlar arasında etkileşimi sağlayan**, kullanıcıya ve sonradan yazılacak programlara birtakım hazır imkanlar sunan, donanım olsun yazılım olsun sistem kaynaklarını paylaştıran ve yöneten **yazılım ağırlıklı bir sistemdir**. DOS, Windows, UNIX, LINUX bilinen işletim sistemleridir.
- **Karakter Tablosu nedir?** : Bilgisayar ortamında metinleri, yazıları oluşturmak için kullanılan **çizelgelerdir**. Bellekte sayılar tutulur; ancak bir metin, harflerin, bir dilin gramer yapısına göre anlamlı bir şekilde sıralanmasıyla oluşur. Dolayısıyla bir metni bilgisayar belleğinde saklama için harf ve sayı dönüşümü yapılan çizelgeler tutulur. **ASCII Karakter tablosuna** göre “BABA” kelimesi bilgisayarda 65 66 65 66 olarak saklanır. Bilgisayar dünyasında en çok kullanılan karakter tablosu ASCII ve Unicode olarak adlandırılır. (Bu kodların hepsini ezberlemek zaten imkânsızdır (: Bu sebeple büyük ‘A’ ve küçük ‘a’yı bilmeniz yeterlidir. A=65 a=97. Örneğin küçük ‘c’yi istiyorsunuz. O zaman a=97 b=98 c=99. Sırayla gittiği için raht bir şekilde bulabilirsiniz.
- **Veri Yapısı nedir?** : Bilginin bellekte tutulma şeklini ve düzenini gösterir. Programlama dillerinde, genel olarak, tamsayı, kesir sayı, karakter ve sözcük saklanması için veri yapıları vardır. Veri yapısı verinin bilgi olmasını sağlar; aynı veri farklı veri yapılarında farklı bilgi olabilir.

2.) C nedir? Ne işe yarar? Nerden çıktı? Nasıl öğrenirim? C için kaynaklar?

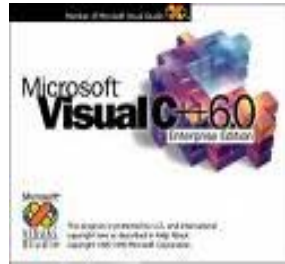
C dilinin oluşmasını UNIX işletimi sağlamıştır. Bu sisteme uygun bir dil kullanmak isteyen **Dennis Ritchie Bell Laboratuvarlarında C dilini oluşturdu**. 1978 yılında ise **“The C Programming Language”** isimli yayınıyla **tanındı**. UNIX sisteminin daha çok kullanılmasıyla zamanla popüler olmuştur. Üzerinden onca zaman geçmesine yeni diller bulunmasına rağmen hala temel olarak gösterilmekte, kullanılmakta. Eğer günümüzde programlamaya başlıyor iseniz C yazmadan bunu gerçekleştiremezsiniz. Programlamanın temeli olan bu dili öğrenmek, kurallarını benimsemek ve çok örnekle kendimizi geliştirmeliyiz.

C Dilinin Avantajları

- C Programlama Dili'ni popüler kılan önemli nedenler aşağıda listelenmiştir: C, güçlü ve esnek bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafik çizebilirsiniz.
- C, iyi bir yazılım geliştirme ortamına sahiptir.
- C, özel komut ve veri tipi tanımlamasına izin verir.
- C, taşınabilir bir dildir.
- C, gelişimini tamamlamış ve standardı oluşmuş bir dildir.
- C, yapısal bir dildir. C kodları *fonksiyon* olarak adlandırılan alt programlardan oluşmuştur.
- C++, Java, JavaScript, JavaApplet, PHP, C#, ... gibi diller C dilinden esinlenmiştir.

C / C++ Derleyicileri

- En sık kullanılan C derleyicileri TURBO C, DEV C++, MICROSOFT VISUAL C++ dır.



1. [Dev-C++](#) 2. [Salford \(Silverfrost\)](#) 3. [GCC \(GNU Compiler Collection\)](#)
4. [Turbo C 2.01](#) 5. [Eclipse IDE](#) 6. [NetBeans IDE](#)

Dev-C++

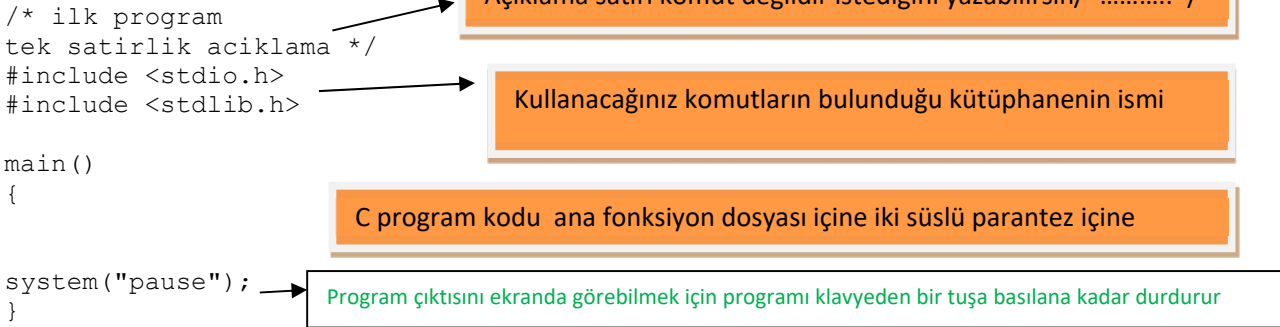
- Derleyicinin ilk üretildiği tarih: 1991-2009
- İndirme Konumu:
<http://www.bloodshed.net/dev/devcpp.html>



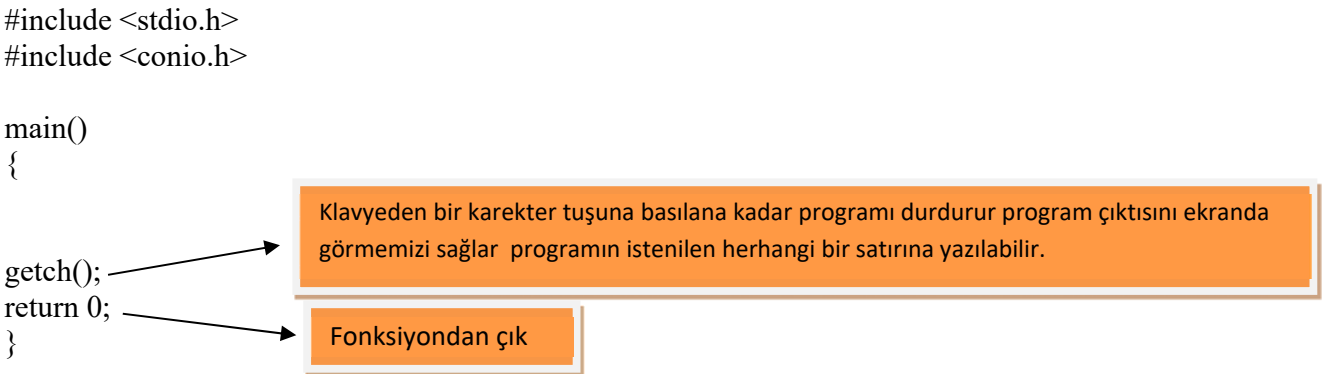
C Kütüphaneleri

- <stdio.h> standart giriş/çıkış işlemlerine izin verir.
- <string.h> String fonksiyonlarını içerir
- <math.h> Matematik fonksiyonlarını içerir.
- <conio.h> Klavye ve ekran kullanımı için gerekli fonksiyonları barındırır. Örneğin getch()
- <stdlib.h> [min](#), [max](#) gibi iki makronun, [exit](#), [failure](#) gibi standart birkaç değer, bazı ana limitlerin, kalanlı bölüm [structure](#)'larının tanımını, ve birkaç standart fonksiyonun prototipini içerir.

KALIP 1



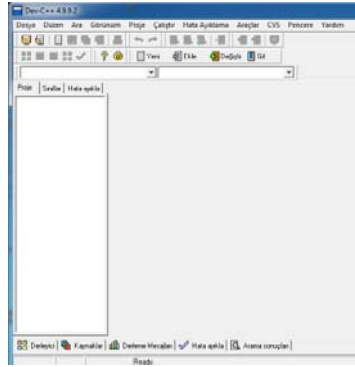
KALIP 2



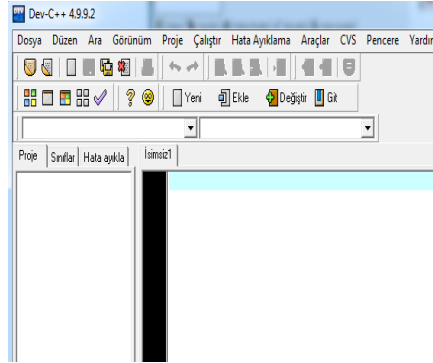
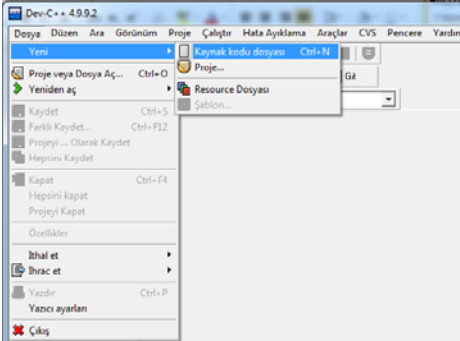
ÖRNEK 1

```
/* ilk program
tek satirlik aciklama */
#include <stdio.h>
#include <stdlib.h>
main()
{
printf("ilk prgram!");
system("pause");
}
```

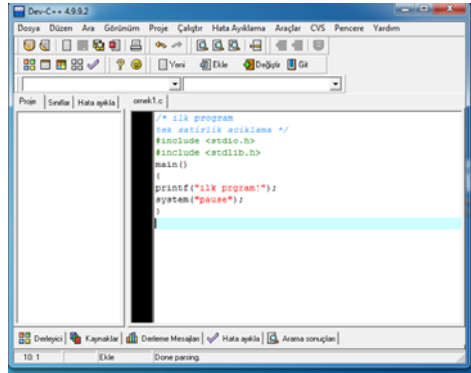
1...Dev++ derleyicisini çalıştır



2.. Yeni bir kaynak kodu dosyası aç

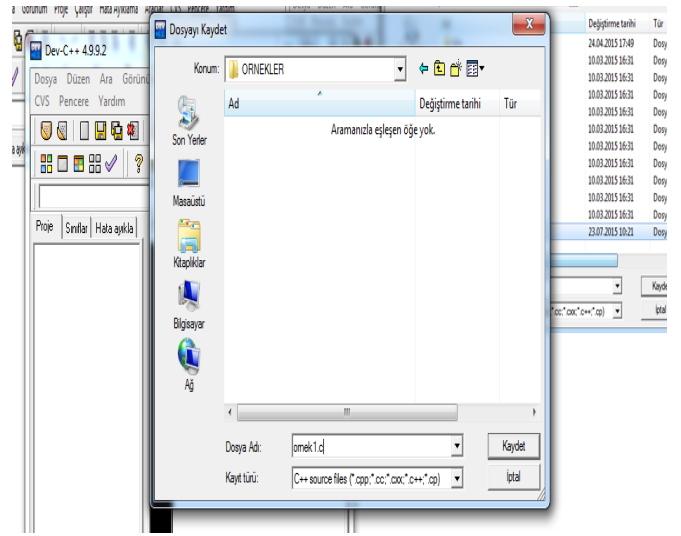
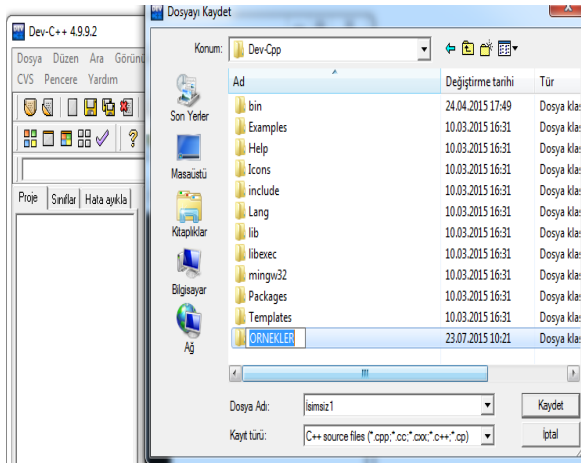


3.. C Program kodunu yaz



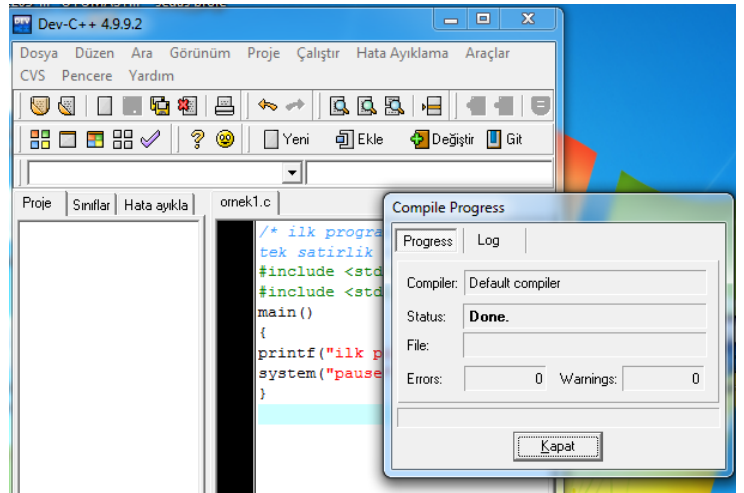
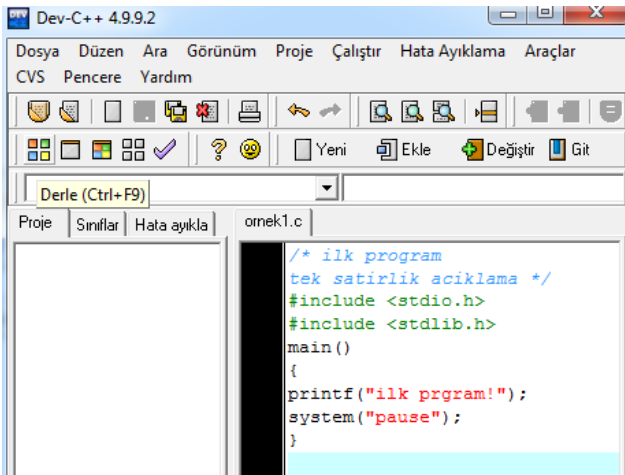
4... Kaydet ancak hard disk de (c içinde olabilir) DevCpp klasörünü bul onun içine kaydetmen gerekiyor DevCpp klasörünün içine yeni bir klasör aç rastgele istediğin bir ismi ve(ÖRNEKLER gibi) bu klasörün içine

6

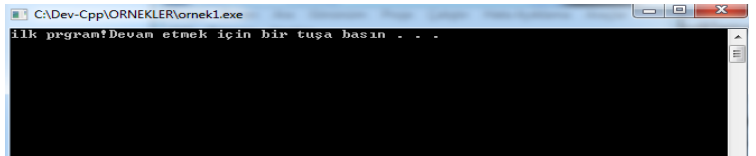
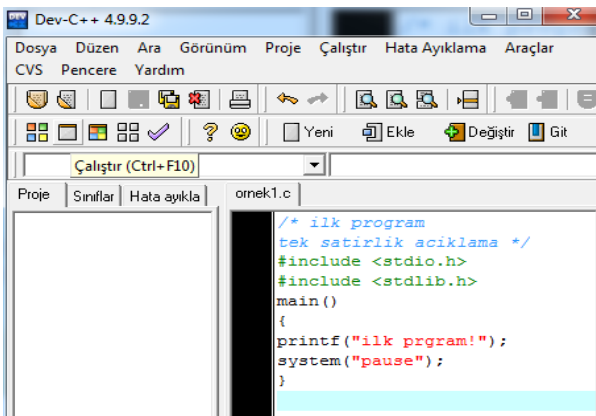


dosyaya bir isim vererek kaydet isim verirken mutlaka .c uzantılı yazmayı unutma ali.c , veli.c , ornek1.c gibi ve kaydet butonuna tıkla

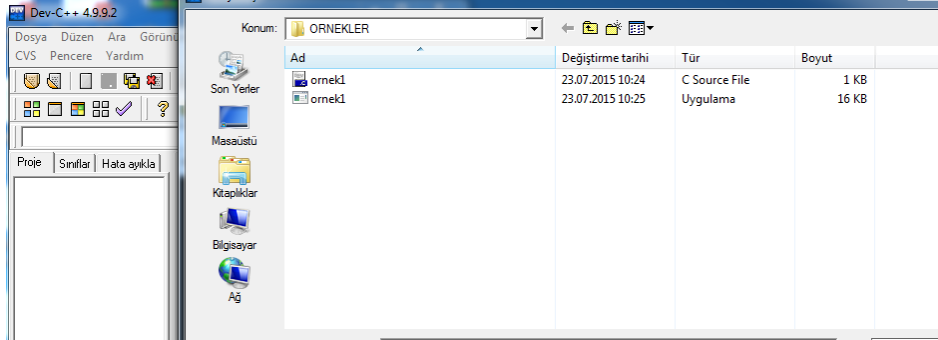
5.... Program kodunu derle (Ctrl+F9 tuşu kısa yoludur) ve Errors (hatalar) 0 olması gerekiyor hata varsa düzelt. Done derleme yaptı mesajını ekranda gördüysen kapat düğmesine tıkla. Derleme yapmak demek bilgisayarın anladığı dil makinadiline 010101 lere program kodunu çevirmek demektir.



6... Programı çalıştır (kısa yolu Ctrl+F10)



Not: yaptığımız örnek program DevCpp klasörü içine açtığımız yeni klasör içinde aşağıda resimde verilen şekilde iki dosya olarak bulunacaktır. Başka bir bilgisayarda programınızı yeniden derlemeye gerek kalmadan iki dosyayı taşıyarak çalıştırabilirsiniz.



Bu kodu yazıp derler ve çalıştırırsanız (F9 bu işlem için programda kolaylık sağlayacaktır. Not: Programı kaydederken ismin altındaki kısımdan mutlaka program türünü C yapmalısınız. İlk zamanlarda çok sorun oluyor (:) ekrana “Merhaba Dünya!” yazısının geldiğini görürsünüz. Şimdi gelelim Sadece 2 kelime için yukarıda yazdıklarımızın ne anlama geldiğine. Bunları tek tek inceleyelim:

// ve /*..*/ Bunlar ne işe yarar? : Bunlar açıklama yazmanız içindir. Derleyici o işaretlerin arasını (Çok satırlık olanı yani 2.si) ya da o işaretin bulunduğu satırı (ilki) göze almaz. Bu kodları istediğiniz yere yazarsınız ve açıklama yaparsınız.

#include <stdio.h> : Program birbirinden bağımsız olan fonksiyonlardan oluşur. #include ise bu programın kapsayacağı fonksiyonların tanımlarının ne iş yaptığının bulunduğu kütüphaneyi bildirir. Bu kod (#include <stdio.h>) standart giriş çıkış dosyalarını programa dahil et yada programın kapsamasını sağla gibisinden bir anlam taşır. Yani program başta derleyiciye ben fonksiyonlar kullanacağım bunların anlamı stdio.h kütüphanesinde var der. Her kütüphanenin fonksiyonları farklıdır. Stdio.h yani “Standart In Out” kütüphanesi temel fonksiyonları içerir (Örnek printf(); scanf(); vs.). Kullandığım logo’nun anlamı da bir bakıma ortaya çıkmıştır :D Bunun dışında C ve öğrendiğim kadarıyla kütüphaneyi dahil etme olayı hemen hemen tüm diğer programlama dillerinde olağan bir durumdur.

main() : Özel bir fonksiyon olup ana program anlamındadır. Ya da ana fonksiyondur diyebilirsiniz. Program ilk başlanıldığında bu fonksiyonun başından başlanır. (Daha detay için

printf(“Merhaba Dünya!”); : Bu fonksiyon belirttiğimiz gibi yazdırma fonksiyonudur. Tırnak işaretleri arasındaki her şeyi ekrana yazdırır. Dikkat ettiyseniz fonksiyonun sonuna genel adı içinde olmamasına rağmen noktalı virgül koyduk. Noktalı virgülü her komuttan sonra kullanmak zorundayız. Çünkü C dilinde noktalı virgül komut ayırıcı olarak kullanılmaktadır.

system(“pause”); : Bu fonksiyonu yazmazsanız program yine çalışır. Fakat siz daha ne olduğunu anlamadan kapanır. Sonuçta program sizden istenileni yapmış ekrana yazınızı yazmış ve kapanmıştır. Bunu engellemek için adından da anlayabileceğiniz gibi bu durdurma kodunu yazıyoruz ve ekran bir tuşa basana kadar açık kalıyor. (Alternatif olarak “return 0;” fonksiyonunu kullanabilirsiniz. İlerde bu fonksiyonu daha detaylı görebileceğiz.)

İlk kullanacağımız ise “printf();” fonksiyonudur. Bu fonksiyon adından da anlayabileceğiniz gibi ekrana yazdırma fonksiyonudur.

ÖRNEK 1.

```
/* ilk program
tek satirlik aciklama */
#include <stdio.h>
#include <stdlib.h>
main()
{
printf("ilk prgram!");
system("pause"); }
```

```
#include <stdio.h>
//include <stdlib.h>
#include <conio.h>
int main(){
printf("Merhaba Dunya!\n");
printf("Ben Ali!\n");
// printf("Merhaba Dunya!\nBen Ali!\n");
//printf("Ali \"Merhaba Dunya!\" dedi.\n");
//system("pause");
getch();
return 0;}
```

ÖRNEK 2.**TEMEL GİRİŞ ÇIKIŞ İŞLEMLERİ****Değişken Tanımlama ve
Değişkene Değer Atama****Değişken tanımları:**

int a; tamsayı türünde a adında bir değişken
float b; ondalıklısayı
char c; karakter

char veri tipi aslında 8 bit tamsayı saklar. 'A' karakteri için istersek ASCII karşılığı olan 65 kullanabiliriz.

Değişkenlere değer atanması:

a = 5; b = 3.75; c = 'A'; c = 65;

NOT: Değişken ismi solda, atanacak değer sağda olmalıdır (5 = a; doğru değildir):

a = b; a değişkenine b değişkenindeki değeri atar
b = a; b değişkenine a değişkenindeki değeri atar

TİP	DEKLARASYON	printf();	scanf();	Minimum	Maksimum	Byte
Karakter	char degisken;	printf("%c",degisken);	scanf("%c",°isken);	-128	127	1
Kısa Tam Sayı	short degisken;	printf("%d",degisken);	scanf("%d",°isken);	-32768	32767	2
Tamsayı	int degisken;	printf("%d",degisken);	scanf("%d",°isken);	-32768	32767	2
Uzun Tamsayı	long int degisken;	printf("%ld",degisken);	scanf("%ld",°isken);	-2147483648	2147483647	4
İşaretsiz Tamsayı	unsigned int degisken;	printf("%u",degisken);	scanf("%u",°isken);	0	65535	2
İşaretsiz Uzun Tamsayı	long unsigned degisken;	printf("%lu",degisken);	scanf("%lu",°isken);	0	4294967295	4
Virgüllü Sayı	float degisken;	printf("%f",degisken);	scanf("%f",°isken);	1,17549e-38	3,40282e+38	4
Uzun Virgüllü Sayı	double degisken;	printf("%lf",degisken);	scanf("%lf",°isken);	2,22504e-308	1,79769e+308	8

Karakter	Anlamı
\a	Ses üretir (alert)
\b	imleci bir sola kaydır (backspace)
\f	Sayfa atla. Bir sonraki sayfanın başına geç (formfeed)
\n	Bir alt satıra geç (newline)
\r	Satır başı yap (carriage return)
\t	Yatay TAB (horizontal TAB)
\v	Dikey TAB (vertical TAB)
\"	Çift tırnak karakterini ekrana yaz
\'	Tek tırnak karakterini ekrana yaz
\\	\ karakterini ekrana yaz
%%	% karakterini ekrana yaz

Sabit (Constant)

- Değişkenden farkı; tanımlandığı anda atanan değerinin program içinde değiştirilememesidir.
- Tanımlanması değişken tanımına benzer. Sadece önüne const eklenir.

`const float PI = 3.142857;`

- Programda PI sabitine atanan ilk değeri sonraki satırlarda kullanabiliriz ama değiştiremeyiz.

`float A = PI*2; doğru`
`PI = 3.15; hatalı`

Veri tipi belirteçleri

%d	İşaretli onluk tabanda sayı
%u	İşaretsiz onluk tabanda sayı
%c	Karakter
%s	Karakter dizisi
%o	İşaretsiz sekizlik tabanda sayı
%x	İşaretsiz onaltılık tabanda sayı
%f	Kayan noktalı (float)
%lf	Kayan noktalı (double)

printf ile formatlı yazdırma:

Değişken tanımları: `int x = 123456;`
 `float y = 22.0 / 7.0;`

`printf ("%10d\n", x);`

10 hanelik yere rakamlar sağa dayalı olarak yazılır. 6 haneli rakam yazılacağından ilk 4 hane boş kalacaktır.

Ekran çıktısı: [123456]

`printf ("% -10d\n", x);`

'-' işaretinden dolayı sağa değil sola dayalı yazılır. Dolayısı ile son dört hane boş kalacak: [123456]

printf ile formatlı yazdırma:

`printf ("%2d\n", x);`

123456 sayısı 6 haneli olduğu için 2 haneye sığmaz, dolayısı ile tamamı yazılır. Ekran çıktısı: [123456]

`printf ("%+-10d\n", x);`

Rakamın başına '+' işareti eklenir. Artı işaretinden dolayı son 3 hane boş kalacaktır: [+123456]

`printf ("%9.3f\n", y);`

9.3f: 9 hanelik yere noktadan sonra 3 hane olacak şekilde sağa dayalı olarak yaz. Nokta işareti de bir hane yer kaplayacak, dolayısıyla ilk 4 hane boş kalacak: [3.143]

ÖRNEK 3.

```
//Değişken tanımlama.
#include <stdio.h>
#include <stdlib.h>
main() {
    int x;
    float y;
    x=1;
    y=1.6;
    printf("X sayisi = %d \n",x);
    printf("Y sayisi = %f \n",y);
    printf("Y sayisi = %.2f \n",y);
    system("pause");
}
```

ÖRNEK 4.

```
//Değişkenlerle işlemler.
#include <stdio.h>
#include <stdlib.h>
main() {
    int x,y,z;
    float sonuc1,sonuc2,sonuc3;
    printf("X,Y ve Z degerleri giriniz : ");
    scanf("%d %d %d",&x,&y,&z);
    sonuc1=x+y;
    printf("X + Y isleminin sonucu : %f \n",sonuc1);
    // "Sonuc 1" float oldugundan %f kullanıldı.
    sonuc2=(x+y)*z;
    printf("(X + Y)*Z isleminin sonucu : %f \n",sonuc2);
    sonuc3=((x*z)*z)+y;
    //İşlemleri küme parantezleriyle ayırıyoruz.
    printf("(X*Z)*z+y isleminin sonucu : %f \n",sonuc3);
    system("pause");}
```

ÖRNEK 5

```
/* Değişkenler ve sabitlerin ekrana yazdırılması */
#include <stdio.h>
#include <stdlib.h>
#define PI 3.141593
int main()
{
    const int MAX = 100;
    char c = 'a';
    char *s = "Bu bir metin";
    int i = 22;
    float f = 33.3;
    double d = 44.4;
    printf("PI = %lf\n",PI);
    printf("MAX= %d\n", MAX);
    printf("c = %c\n", c);
    printf("s = %s\n", s);
    printf("i = %d\n", i);
    printf("f = %f\n", f);
    printf("d = %lf\n",d);
    system("pause");
    return 0;}
```

ÖRNEK 6

```
//Scanf kullanma.
#include <stdio.h>
#include <stdlib.h>
main() {
    int x;
    printf("Bir deger giriniz : ");
    scanf("%d",&x);
    printf("Girdiginiz deger : %d \n",x);
    system("pause");}
```

Yukarıdaki örnekte ilk başta programa x değişkenini int olarak tanımladık. Daha sonra ekrana yazdırma fonksiyonu ile kullanıcıdan değer istedik. scanf fonksiyonunda tırnak işaretleri arasındaki %d ya da değişkeninize göre %f %c koyduktan sonra tırnak işaretini kapatıp &x yazarak girilen değeri x'e tanımlıyoruz hepsi bu kadar. Eğer birden fazla değer girilecek olsaydı bu işlemi tekrarlayabilir ya da aşağıdaki örnekteki gibi programınızı düzenleyebilirdiniz:

ÖRNEK 7

//Scanf kullanma.

```
#include <stdio.h>
#include <stdlib.h>
main() {
    int x,y;
    printf("X ve Y icin deger giriniz : ");
    scanf("%d %d",&x,&y);
    printf("X = %d ve Y=%d \n",x,y);
    system("pause");}
```

ÖRNEK 8 Bir karenin kenarı girildiğinde alanı ve çevresini hesaplayan program

```
#include <stdio.h>
#include <stdlib.h>
main () {
    int a;
    printf("Karenin kenar uzunlugunu girin : ");
    scanf("%d",&a);
    printf("Karenin Alani : %d \n",a*a);
    printf("Karenin kenar uzunluklari toplami : %d \n",4*a);
    system("pause");}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Yukarıdaki örnekte iki int tanımladık. Daha sonra kullanıcıdan bu değişkenler için değer istedik. Üsttekiyle pek farkı yok aslında. Önemli olan nokta ise scanf'de tırnak işareti içindeki değişkenler nasıl gösterilmiş ise kullanıcı da değişkenleri o şekilde girmek zorundadır. Yukarıdaki örneğe göre kullanıcı ekrana şu şekilde giriş yapmalı: "1 2". scanf'in içi şu şekilde olsaydı: "%d,%d" kullanıcı yine o şekilde yani "1,2" yazmak zorundaydı.

puts () Fonksiyonu

Ekrana yazdırılacak ifade bir karakter topluluğu ise, printf() 'e alternatif olarak puts() fonksiyonu kullanılabilir. Ancak puts(), ekrana bu karakter topluluğu yazdıktan sonra, imleci alt satıra geçirir. Buna göre:

```
printf("Sevgi varlığın mayasıdır.\n");
ile
puts("Sevgi varlığın mayasıdır.");
kullanımları eşdeğerdir.
```

puts() fonksiyonu Tabloda verilen \n \a gibi kontrol karakterleri ile kullanılabilir.

```
puts("Bu birinci satır...\nBu ikinci satır.");
```

gets () Fonksiyonu

Klavyeden bir karakter topluluğu okumak için kullanılır. Okuma işlemi yeni satır karakteriyle(\n) karşılaşıncaya kadar sürer. puts() - gets() arasındaki ilişki, printf() - scanf() arasındaki gibidir. Yani,

```
scanf("%s",str);
ile
```

```
gets(str);
aynı anlamdadır.
```

getchar () Fonksiyonu

Bu fonksiyon ile standart girişten bir karakter okunur. Programı istenen bir yerde durdurup, bir karakter girinceye kadar bekletir. Örneğin:

```
...
```

```

for(i=0; i<10; i++)
{
    getchar();
    printf("%d\n",i);
}
...

```

Yukarıdaki program parçası 0-9 arası sayıları sırasıyla ekranda göstermek için kullanılır. Fakat her rakamı yazdırılmadan önce klavyeden herhangi bir karakter girip [Enter] tuşuna basılması beklenir. Bu bekleme `getchar()` fonksiyonu ile gerçekleştirilir.

Atama Operatörleri

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

Bu tanımlamalara göre, aşağıdaki atamaları inceleyiniz:

```

/* bir arttırma işlemleri */
i++;
++i;
i += 1;
i = i + 1;

/* karmaşık atamalar */
f *= i;      // f = f * i; anlamında
f *= i+1;    // f = f * (i+1); anlamında
z /= 1 + x;  // z = z / (1+x); anlamında

```

Örnek 9

```

#include <stdio.h>
#include <stdlib.h>
main () {
    int a,b,c;
    a=5;a=a+1;printf("\na= %d",a);
    b=5;b++;printf("\nb= %d",b);
    c=5;c--;printf("\n c= %d",c);
    system("pause");}

```

Bazı Trigonometrik fonksiyonlar

Fonksiyon	Açıklama	Örnek
<code>double sin(double x)</code>	x sayısının sinüs değerini radyan cinsinden hesaplar	<code>y = sin(0.22)</code>
<code>double cos(double x)</code>	x sayısının kosinüs değerini radyan cinsinden hesaplar	<code>y = cos(0.14)</code>
<code>double tan(double x)</code>	x sayısının tanjant değerini radyan cinsinden hesaplar	<code>y = tan(0.82)</code>

ÖRNEK 10:::

/ 30 dercelik acinin sinus, kosinus, tanjant ve kotanjant degerleri */*

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.141593
int main()
{
    double aci = 30.0;
    aci *= PI/180.0; /* radyana ?evir */
    puts("30 derecenin");
    printf("sinusu : %lf\n", sin(aci));
    printf("kosinusu : %lf\n", cos(aci));
    printf("tanjanti : %lf\n", tan(aci));
    system("pause");
    return 0;}
```

pow() - sqrt() ve logaritmik fonksiyonlar

Fonksiyon	Açıklama	Örnek
double pow(double x, double y)	x^y değerini hesaplar	$\text{pow}(2.0, 3.0) = 2^3 = 8$
double sqrt(double x)	pozitif x sayısının karekökünü hesaplar	$\text{sqrt}(4.0) = 4^{1/2} = 2$
double log(double x)	pozitif x sayısının doğal logaritmasını hesaplar, $\ln(x)$	$\log(4.0) = 1.386294$
double log10(double x)	pozitif x sayısının logaritmasını hesaplar	$\log_{10}(4.0) = 0.602060$

ÖRNEK 11::

/ pow(), sqrt(), log() ve log10() fonksiyonlarının kullanımı */*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
main()
{
    double x, y, z;
    x = 64.0;
    y = 3.0;
    z = 0.5;
    printf("pow(64.0, 3.0) = %7.0f\n", pow(x,y) );
    printf("sqrt(64.0) = %2.0f\n", sqrt(x) );
    printf("pow(64.0, 0.5) = %2.0f\n", pow(x,z) );
    printf("ln(3.0) = %f \n", log(y) );
    printf("log(3.0) = %f \n", log10(y) );
    system("PAUSE"); return 0; }
```

ÖRNEK 12:::

%%%

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
main()
{
    float sayi; printf("lutfen bir sayi giriniz"); scanf("%f",&sayi);
    printf("\ngirmis oldugunuz sayinin karakoku %.2f",sqrt(fabs(sayi)));
    printf("\ngirmis oldugunuz sayinin exp degeri %.2f",exp(sayi));
    printf("\ngirmis oldugunuz sayinin logaritmik degeri %.2f",log(sayi));
    printf("\ngirmis oldugunuz sayinin tavan degeri %.2f",ceil(sayi));
    printf("\ngirmis oldugunuz sayinin taban degeri %.2f",floor(sayi));
    system("PAUSE");
    return 0; }
//abs(x) int olan x in mutlak degeri ,
// fabs(x) float olan x in mutlak degeri
//cabs(x) complex olan x in mutlak degeri
```


KARMAŞIK SAYILAR

ÖRNEK 13:::

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#include <stdio.h>
#include <complex.h>
#include <stdlib.h>
int main()
{
    double complex z1 = 1.0 + 3.0 * I;
    double complex z2 = 1.0 - 4.0 * I;
    printf("Working with complex numbers:\n\v");
    printf("Starting values: Z1 = %.2f + %.2fi\tZ2 = %.2f %+.2fi\n", creal(z1), cimag(z1), creal(z2), cimag(z2));
    double complex sum = z1 + z2;
    printf("The sum: Z1 + Z2 = %.2f %+.2fi\n", creal(sum), cimag(sum));
    double complex difference = z1 - z2;
    printf("The difference: Z1 - Z2 = %.2f %+.2fi\n", creal(difference), cimag(difference));
    double complex product = z1 * z2;
    printf("The product: Z1 x Z2 = %.2f %+.2fi\n", creal(product), cimag(product));
    double complex quotient = z1 / z2;
    printf("The quotient: Z1 / Z2 = %.2f %+.2fi\n", creal(quotient), cimag(quotient));
    double complex ali = cabs(z1);
    printf("z1 in karelerinin kare koku %.2f ", ali);
    double complex veli = carg(z1);
    printf("\nz1 in argumani radyan olarak::: ters tanjan imag/real %.2f ", veli);
    printf("\nz1 in argumani derece olarak::: ters tanjan imag/real %.2f ", veli*(180/3.14));
    getch(); return 0;}

```

ÖRNEK 14:::

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#include <stdio.h> /* Standard Library of Input and Output */
#include <complex.h> /* Standart Library of Complex Numbers */
#include <stdlib.h>
int main()
{
    double complex z1 = 1.0 + 3.0 * I;
    double complex z2 = 1.0 - 4.0 * I;
    printf("Working with complex numbers:\n\v");
    printf("Starting values: Z1 = %.2f + %.2fi\tZ2 = %.2f %+.2fi\n",
        creal(z1),
        cimag(z1),
        creal(z2),
        cimag(z2));
    double complex sum = z1 + z2;
    printf("The sum: Z1 + Z2 = %.2f %+.2fi\n", creal(sum), cimag(sum));
    getch();}

```

IF ŞARTI

```

if (koşul)
{
    ...
    deyimler; (küme)
    ...
}

```

Koşul değimi doğru (1) yada yanlış (0) değeri üretir. Şartın doğru olması durumunda if satırından sonraki deyimler icra edilir. Şartın yanlış değeri üretmesi durumunda else den sonraki deyimler icra edilir.

```

/* 15.) if deyiminin kullanımı */
#include <stdio.h>
#include <conio.h>
main() {
int x, y;
    printf("\nBir tamsayı değeri girin, x: ");
    scanf("%d", &x);
    printf("\nBir tamsayı değeri girin, y: ");
    scanf("%d", &y);
    if (x == y)
        printf("x, y ye eşit\n");
    if (x > y)
        printf("x, y den büyük\n");
    if (x < y)
        printf("x, y den küçük\n");
    getch();
    return 0;
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

if (koşul)
{
    ...
    deyimler; (küme1)
    ...
}

else
{
    ...
    deyimler; (küme2)
    ...
}

```

```

/*16.) Klavyeden girilen bir sayının çift olup olmadığını sınırlar.*/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int sayi;
    printf("Bir sayi girin: ");
    scanf("%d",&sayi);
    if (sayi % 2 == 0)
        printf("sayi cifttir.\n");
    else
        printf("sayi tektir.\n");
    system("pause"); return 0;}

```

Mantıksal Operatörler kullanarak birden çok karşılaştırma birleştirilebilir. Buna iyi bir örnek 4 de gösterilmiştir. Program, bir yılın artık yıl olup olmadığını sınırlar. Bir yıl içinde, Şubat ayı 29 gün olursa o yıl artık yıl olarak adlandırılır. Artık yıl periyodik olarak 4 yılda bir gelir. Genel kanı *"bir yıl 4 ile tam bölünebilirse o yıl artık yıldır"* şeklindedir. Fakat 1996 artık yıl iken 1800 artık yıl değildir. Genel sorgulama şöyle olmalıdır: Eğer bir yıl 4 ile tam bölünüyorsa VE 100'e tam bölünmüyorsa VEYA 400'e tam bölünüyorsa o yıl artık yıldır.

```

/*17 Bir yılın artık yıl olup olmadığını sınırlar.*/
#include <stdio.h>
#include <stdlib.h>
void main(){
    int yıl;
    printf("Bir yıl girin: ");
    scanf("%d",&yıl);
    if( yıl % 4 == 0 && yıl % 100 != 0 || yıl % 400 == 0 )
        printf("%d artık yıl\n",yıl);
    else
        printf("%d artık yıl değil\n",yıl);
    system("pause");}

```

```

if(koşul)
{
    ...
    deyimler;(küme1)
    ...
}

else if
{
    ...
    deyimler;(küme2)
    ...
}

else if
{
    ...
    deyimler;(küme3)
    ...
}

.
.

```

```

else
{
...
deyimler;(kümeN)
...
}

```

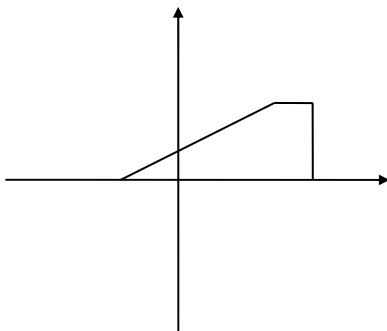
/*18 Diskriminant hesabı. $ax^2 + bx + c = 0$ denkleminin,karmaşık sayılı kökler dahil, çözümü. */

```

#include <stdio.h>
#include <math.h> /* sqrt için */
#include <stdlib.h>
float a,b,c,delta,x1,x2,x,kok_delta;
main() {
printf("a, b, c değerlerini girin:\n");
scanf("%f %f %f",&a,&b,&c);
delta = b*b - 4*a*c;
if( delta > 0.0 )
{
x1 = (-b + sqrt(delta))/(2*a);
x2 = (-b - sqrt(delta))/(2*a);
printf("\nReel kökler :");
printf("\nx1 = %f ve x2 = %f",x1,x2);
}
else if( delta < 0.0 )
{
kok_delta = ( sqrt(-delta) ) / (2*a);
x = -0.5*b/a;
printf("\nKarmaşık kökler :");
printf("\nx1 = %f + %fi ve x2 = %f - %fi",x,kok_delta,x,kok_delta);
}
else
{
x = -0.5*b/a;
printf("\nKökler eşit :");
printf("\nx1 = x2 = %f",x);
}
system("pause"); return 0; }

```

19) Fonksiyonun girilen t değeri için aldığı değeri hesaplayıp yazan program
((0,-1),(2,0),(3,0)koordinantlarında değişim gösteren aşağıdaki şekilde verilen fonksiyon için)



```

#include <stdio.h>
#include <conio.h>
int main()
{
float y, x;
printf("x değerini giriniz");
scanf("%f", &x);
if ((x > -1) && (x < 2))
y = 2/3 * (x + 1);
else
if ((x > 2) && (x <= 3))
y = 2;
else
y = 0;
printf("\n Fonksiyonun değeri = &f",y);
system("pause");
}

```

20 Örnek 1 girilen sayının 3 ile bölünebileceğinin kontrolü

```
#include<stdio.h>
#include<conio.h>
main(){
int x,y,z,t;
t=0;
printf("birler basamagindaki sayiyi giriniz");
scanf("%d",&x);
printf("onlar basamagindaki sayiyi giriniz");
scanf("%d",&y);
printf("yuzler basamagindaki sayiyi giriniz");
scanf("%d",&z);
t=x+y+z;
if(t%3==0){
printf("\n girilen sayi 3 e tam bolunur");
}
else
{
printf("\n girilen sayi 3 e tam BOLUNMEZ");
}
getch(); return 0 ;}
```

Örnek 21: Girilen üç sayıdan en küçüğünün bulunması

```
#include <stdio.h>
#include <conio.h>
main(){
    int s1,s2,s3,ek;
    scanf("%d%d%d",&s1, &s2,
    &s3);
    if ((s1<s2) && (s1<s3))
    ek =s1;
    else
    if (s2<s3)
    ek =s2;
    else
    ek = s3;
    printf("En küçük olanı = %d", ek);
    getch();
    return 0 ;
}
```

Örnek 22

```
#include <stdio.h>
#include <conio.h>
void main(){
int a,b,i;
printf("ilk sayiyi girin : "); scanf("%d", &a);

printf("ikinci sayiyi girin : "); scanf("%d", &b);

printf("1)toplama, 2)cikarma, 3)carpma, 4)bolme");

printf("islemi secin [1-4]: ");

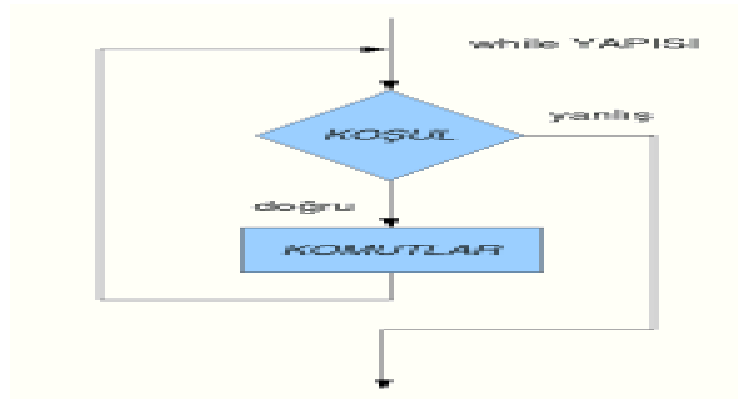
scanf("%d", &i);

if (i == 1) printf("sayilarin toplami : %d\n", a+b);
else if (i == 2) printf("sayilarin farki : %d\n", a-b);
else if (i == 3) printf("sayilarin carpimi : %d\n", a*b);
else if (i == 4) printf("sayilarin bolumu : %d\n", a/b);

getch(); return 0;}
```

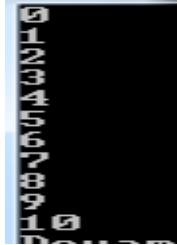
While döngüsü

```
while( koşul ) {
    komut(lar)
}
```



23.)1 DEN 10 kadar sayıları yazdırma

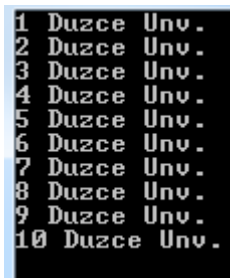
```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=0;
    while(x <= 10)
        printf("%d\n",x++);
    system("pause");
    return 0;
}
```



25) 10 defa duzce unv yazdırma

```
#include<stdio.h>
#include<stdlib.h>
main(void){
    int i=0;
    while(i++<10)
    {
        printf("%d Duzce Unv.\n",i);
    }
    getch();return 0;}

```



24.)Örnek $\sum_{i=0}^n i^2$ hesaplayn program

```
#include<stdio.h>
#include<conio.h>
int main( void )
{
    int i = 0, toplam_deger = 0;
    int n;
    printf("Lütfen n değerini giriniz> ");
    scanf("%d",&n);
    while( i <= n ) {
        toplam_deger += i*i;
        i++;
    }
    printf("Sonuç:
    %d\n",toplam_deger);
    getch(); return 0;}

```

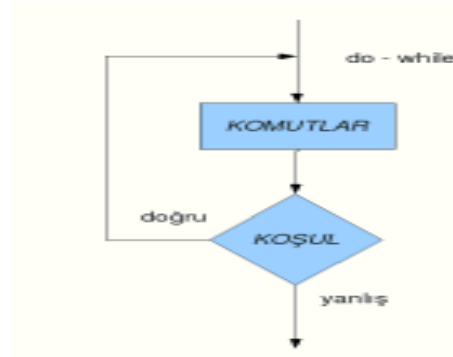
do while Döngüsü

Yaptığı iş, while ile hemen hemen aynıdır; verilen işi, döngü koşulu bozulana kadar sürdürür. Ancak **while'a göre önemli bir farkı vardır.** while döngülerinde, döngü içersindeki işlem yapılmadan önce, sunulan koşul kontrol edilir. Şayet koşul sağlanmıyorsa, o while döngüsünün hiç çalışmama ihtimali de bulunmaktadır. do while döngülerindeyse, durum böyle değildir. İlk çalışmada koşul kontrolü yapılmaz. Dolayısıyla, her ne şartta olursa olsun, döngünüz -en azından bir kere- çalışacaktır.

Bazı durumlarda, döngü bloğu içerisindeki kodların en azından bir kere çalışması gerektiğinden, do while yapısı kullanılır. do while ile ilgili genel yapıyı ve akış şemasını aşağıda bulabilirsiniz:

do while Akış Diyagramı

```
do {
    komut(lar)
} while( koşul );
```



26.)negatif sayı girene kadar 2 katını yazan program

```
#include <stdio.h>
#include <conio.h>
main()
{
    int sayi;
    do
    {
        printf("Bir sayi girin : ");
        scanf("%d",&sayi);
        printf("iki kati : %d\n",2*sayi);
    }while( sayi>0 ); /* koşul */
    puts("Döngü sona erdi.");
    system("pause"); return 0;}
```

```
Bir sayi girin : 2
iki kati : 4
Bir sayi girin : 3
iki kati : 6
Bir sayi girin : -1
iki kati : -2
Döngü sona erdi.
Devam etmek için bir
```

27) 10 defa düzce unv yazan program

```
#include<stdio.h>
#include<stdlib.h>
main(void)
{
    int i=0;
    do
    {
        printf("%2d Duzce Unv.\n",++i);
    }
    while(i<10);
    getch();return 0;}
```

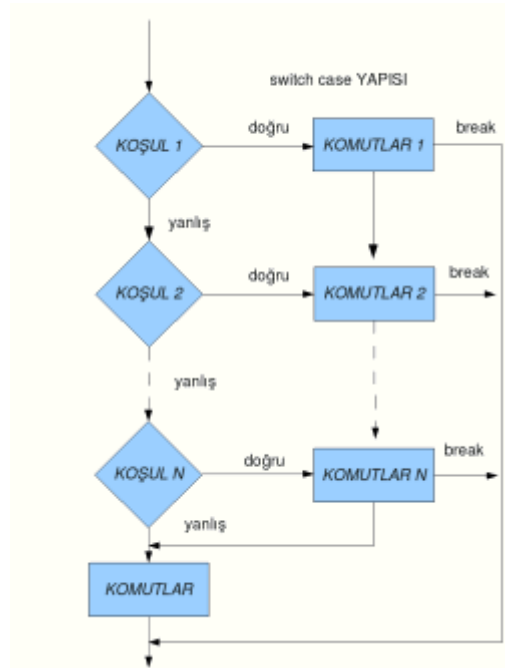
(printf de ++i yerine i yazılırsa Program sonsuz döngüye girer Ctrl c ile çıkılır)

SWITCH CASE YAPISI

switch case Yapısı

```
switch( degisken ) {
    case sabit1:
        komut(lar)
        [break]
    case sabit2:
        komut(lar)
        [break]
    .
    .
    case sabitN:
        komut(lar)
        [break]
    default:
        komut(lar);
}
```

switch case Akış Diyagramı



Yapı olarak şimdiye kadar görmüş olduğunuz if else gibi gözükme de, bir örnekten sonra arasında pek bir fark olmadığını göreceksiniz. Her komut sonunda koyduğum break komutu, zorunlu değildir ve o nedenle köşeli parantezle belirtilmiştir. break koyduğuz takdirde, uygun koşul sağlandıktan sonra, daha fazla kontrol yapılmayacak ve aynen if - else if yapısında olduğu gibi program orada kesilecektir. Ama break koymazsanız, altında kalan bütün işlemler -bir daha ki break'e kadar- yapılacaktır. Kodun sonunda gördüğünüz default komutu, if - else if yapısında ki sonuncu else gibidir. Uygun hiçbir şart bulunamazsa, default komutu çalışır. Bu deyim bir *değişkenin* içeriğine bakarak, programın akışını bir çok seçenektan birine yönlendirir. case (durum) deyiminden sonra değişkenin durumu belirlenir ve takip eden gelen satırlar (deyimler) işleme konur. Bütün durumların aksi söz konusu olduğunda erçekleştirilmesi istenen deyimler default deyiminden sonraki kısımda bildirilir. Genel yazım biçimi:

28).Önce işlem sonra iki sayı gir işlemi yapısın

```
#include<stdio.h>
#include <stdlib.h>
main()
{
    char islem;
    int s1, s2, s3;
    printf("Önce işlemi sonra sayıları girin ");
    scanf("%c%d%d",&islem, &s1, &s2);
    switch (islem) {
        case '+' : s3 = s1 + s2; break;
        case '-' : s3 = s1 - s2; break;
        case '*' : s3 = s1 * s2; break;
        case '/' : s3 = s1 / s2; break;
        default : printf ("Hatalı işlem");
    }
    printf("\nSonuç = %d",s3);
    getch();
    return 0;}

```

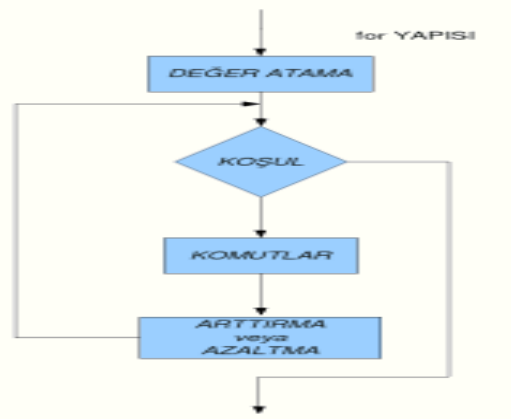

29). yılın kaçınıcı ayında olduğunu girince Mevsimleri yaz.

```
#include<stdio.h>
#include <stdlib.h>
int main( void )
{
    int ay;
    printf("yilin kacinci ayi???:::");
    scanf("%d", &ay);
    switch (ay) {
    case 3:
    case 4:
    case 5: printf("ilkbahar"); break;
    case 6:
    case 7:
    case 8: printf("yaz"); break;
    case 9:
    case 10:
    case 11: printf("sonbahar"); break;
    case 12:
    case 1:
    case 2: printf("kış"); break;
    }
    getch();return 0;}

```

FOR DÖNGÜSÜ

```
for( ilk_deger_atama; koşul;
    arttırma/azaltma ){
    komut(lar)
}
```



Bu deyim, diğer döngü deyimleri gibi bir kümeyi bir çok kez tekrarlamak için kullanılır. Koşul sınaması `while` da olduğu gibi döngüye girmeden yapılır. Bu döngü deyiminde diğerlerinden farklı olarak başlangıç değeri ve döngü sayacına sahip olmasıdır.

30.) 1den 10 kadar sayıları yazdırma

```
#include<stdio.h>
#include<conio.h>
int main(){

int k=0;
for (; k<10; ++k)
{
printf("%d\n", k);
}
getch();return 0;}
```

31.) 10 defa düzce üniv. yazdırma

```
#include<stdio.h>
#include<conio.h>
int main(){

int i;
for( i = 0 ; i < 10; i++)
{
printf("%2d: Duzce Univ.\n", (i+1));
}
getch();return 0;}
```

32.) 10 defa düzce üniv. yazdırma

```
#include<stdio.h>
#include<conio.h>
int main(){

int i;i=0;
for(i<10;){
printf("%2d: Duzce Univ.\n", (i+1));
i = i + 1;
}
getch();return 0;}
```

33.) 1'den 100'e kadar olan sayıların toplamı. (en son j değeri ekrana basılır)

```
#include<stdio.h>
#include<conio.h>
int main(){
int j=0,i=0;
for (i=1; i<=100; i=i+1)
j =j+i;
printf("Toplam %d",j);
getch();return 0;}
```

34)

```
#include<stdio.h>
#include<conio.h>
int main(){
```

```
int j=0,i=0;
for (i=1; i<=100; i=i+1)
{
j =j+i;
}
printf("Toplam %d",j);
getch();return 0;}
```

/* 34.) for döngüsü ile faktoriyel hesabı. */

```
#include <stdio.h>
#include<stdlib.h>
int main() {

long i, n, faktor;
printf("Faktoriyeli hesaplanacak
sayi girin :");
scanf("%ld", &n);
faktor=1;
for(i=1; i<=n; i++){
faktor *= i;
}
printf("%ld! = %ld\n", n, faktor);

getch(); return 0;}
```

35) büyük sayının faktöriyeli

```
#include <stdio.h>
#include<conio.h>
int main(){

double i, n, faktor;
printf("Faktoriyeli hesaplanacak sayi
girin:");
scanf("%lf", &n);
faktor=1;
for(i=1; i<=n; i++){
faktor *= i;
}
printf("%lf! = %lf\n", n, faktor);

getch();return 0;}
```

%%%%%%%%%

36.) faktöriyel alma

```
#include<stdio.h>
#include<conio.h>
main() {

int i,s1;
long t=1;
printf("sayiyi giriniz...");
scanf("%d", &s1);
for (i=s1; i>=1; --i) t*=i;
printf("%d sayisinin faktoriyeli = %ld", s1, t);

getch();return 0;}
```

%%%%%%%%%

37.) Örnek :printf fonksiyonu ile desimal (taban-10) sayılarının nasıl yazdırılacağı bundan e 0-15 arası desimal sayıların Oktal (taban-8) ve Heksadesimal (taban-16) karşılıkları ile printf kullanılarak yazdırılması

```
/* örnek Sayı sistemi
%d : desimal 10 tabanındaki sayı
%o : oktal 8 tabanındaki sayı
%x : hexadesimal 16 tabanındaki sayı (küçük harf)
%X : hexadesimal 16 tabanındaki sayı (büyük harf) */
```

```
#include <stdio.h>
#include <conio.h>
int main(){
int i;
for (i=0; i<16; i++)
printf("%02d %02o %x %X\n", i,i,i,i);
getch(); return 0;}
```

ÇIKTI

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	10	8	8
9	11	9	9
10	12	a	A
11	13	b	B
12	14	c	C
13	15	d	D
14	16	e	E
15	17	f	F

İç içe Geçmiş Döngüler

/* 38.)Üç basamaklı, basamaklarının küpleri toplamı kendisine eşit olan tam sayılara Armstrong sayı denir. Örneğin: $371 = 3^3 + 7^3 + 1^3$. Bu program İç-içe geçmiş 3 döngü ile bütün Armstrong sayıları bulur. */

```
#include <stdio.h>
#include <conio.h>
int main(){
```

```
int a,b,c, kup, sayi, k=1;
```

```
for(a=1; a<=9; a++)
for(b=0; b<=9; b++)
for(c=0; c<=9; c++)
{
sayi = 100*a + 10*b + c; /* sayi = abc (üç basamaklı) */
kup = a*a*a + b*b*b + c*c*c; /* kup = a^3+b^3+c^3 */
//printf("\n%d%d%d",a,b,c);
if( sayi==kup ) printf("%d. %d\n",k++,sayi);
// printf("\n%d%d%d",a,b,c);
}
//printf("\n%d%d%d",a,b,c);
```

```
getch();return 0;}
```

```
çıktı
1. 153
2. 370
3. 371
4. 407
```

39.) Çarpım tablosu. (iç içe döngüler)

```
#include <stdio.h>
#include <conio.h>
main(){
int i,j;
for (i=1; i<=10; i++) {
for (j =1; j<=10; j++)
printf("%4.0d",i*j);
printf("\n");
}
getch();return 0;}
```

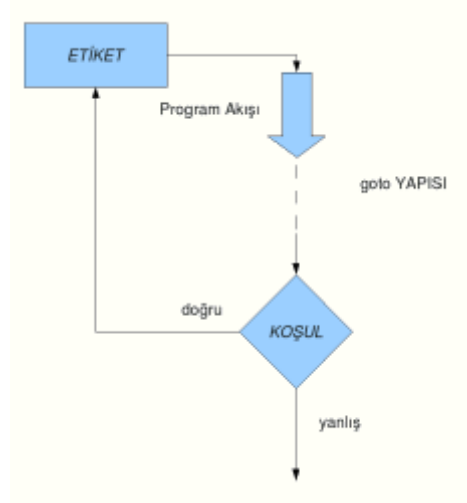
GOTO YAPISI

Birçok programlama dilinde olduğu gibi programın kontrolünün program içinde herhangi bir noktaya transferi mümkündür. etiketler sayesinde, programın bir noktasından bir başka noktasına atlamayı sağlar. goto, bir döngü değildir ancak döngü olarak kullanılabilir. goto, çalışabilmek için etiketlere ihtiyaç duyar. Etiketler, vereceğiniz herhangi bir isme sahip olabilir. Etiket oluşturmak için bütün yapmanız gereken; etiket adını belirleyip, sonuna iki nokta üst üste eklemek (:) ve programın herhangi bir yerine bunu yazmaktır. goto deyimi kullanarak bu etiketleri çağırırsanız, etiketin altında bulunan kodlardan devam edilir. goto ve etiketlere dair genel yapıyı, akış diyagramıyla birlikte aşağıda bulabilirsiniz:

goto Yapısı

```
label_name:
.
.
.
if( kosul )
{
    goto
    label_name
}
.
.
.
```

goto Akış Diyagramı



NOT: goto deyimi tek başına da kullanılabilir. Fakat mantıksal bir sınama olmadan, goto yapısını kullanmanız, sonsuz döngüye neden olacaktır.

40.) %%%%%%%%%%

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

```
#include<conio.h>
```

```
main(){
```

```
int i = 0;
```

başlangic_noktasi:

```
printf( "Duzce Unv\n" );
```

```
i++;
```

```
if( i<10 ) goto baslangic_noktasi;
```

```
getch();return 0;}
```

break Deyimi

işlemin sona erdirilmesi bu deyim ile yapılır. Örneğin, **döngü** deyimleri içindekiler yürütülürken, çevrimin, koşuldan bağımsız kesin olarak sonlanması gerektiğinde bu deyim kullanılır. Bu durumda döngü sonsuzdur. Fakat **döngü içinde if deyimindeki koşul gerçekleşirse, döngü koşuluna bakılmaksızın terkedilir.**

/* 41.) $n \geq 0$ olduğu sürece $n!$ değerini hesaplar */

```
#include <stdio.h>
#include <conio.h>
int main(){

    long int i,n,faktor;
    while(1) /* sonsuz döngü */
    {
        printf("Faktoriyeli hesaplanacak sayı girin : ");
        scanf("%ld",&n);
        if(n<0) break; /* döngüyü sonlandır */
        for(faktor=1, i=1; i<=n; i++)
            faktor *= i;
        printf("%ld! = %ld\n",n,faktor);
    }
    getch();return 0;}
```

continue Deyimi

break komutunun, döngüyü kırmak için olduğundan bahsetmiştik. Bunun dışında işlem yapmadan döngüyü devam ettirmek gibi durumlara da ihtiyacımız vardır. Bunun içinde continue (Türkçe: devam) komutunu kullanırız. Bir döngü içerisinde continue deyimi ile karşılaşırsa, ondan sonra gelen deyimler atlanır ve döngü bir sonraki çevrime girer. Örneğin:

```
...
    for (x=-50; i<=50; x++)
    {
        if(x<0) continue; /* x<0 ise alttaki satırı atla */
        printf("%d\t%f", x, sqrt(x));
    }
...

```

Program parçasının çıktısı:

```
0      0.000000
1      1.000000
2      1.414213
3      1.732050
.      .
.      .
.      .
50     7.071067
```

/*42.) Sadece tek sayıları yazdıran bir program*/

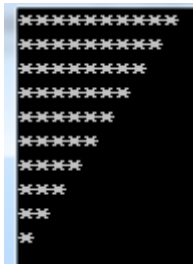
```
#include<stdio.h>
#include<conio.h>
int main( void )
{
    int i;
    for( i = 0; i < 10; i++ )
    {
```

```
        if( i%2 == 0 ) continue;
        printf("%2d\n",i);
    }
    getch(); return 0; }
```

//i değişkeninin 2'ye göre modu
 //0 sonucunu veriyorsa, bu onun
 //bir çift sayı olduğunu gösterir.
 //Bu durumda ekrana yazdırılmaması
 //için döngü bir sonraki adıma geçer.
Çıktısı 1 3 5 7 9

Örnek43:....

```
#include<conio.h>
#include<stdio.h>
main(){
int i,j;
for (i=10;i>=1;i--)
{
for(j=1;j<=i;j++){
printf("*");
}
printf("\n");
}
getch();return 0;}
```

**ÖRNEK 44 defa Duzce Unv yazdırma**

```
#include<stdio.h>
#include<stdlib.h>
main(void){
int i=0;
while(i++<10)
{
printf("%d Duzce Unv.\n",i);
}
getch();return 0;}
```

45 Örnek while ile Faktöriyel alma

```
#include <stdio.h>
#include<conio.h>
main()
{
int i, sonuc;
i = 5;
sonuc = 1;
printf("%d faktöriyel = ", i);
while (i>1)
{
sonuc = sonuc * i;
i--;
}
printf("%d", sonuc);
getch();return 0;}
```

FONKSİYONLAR

Fonksiyonlar C programlama dilinin temel taşlarından sayılırlar. Çalışan kodlarımızı yerleştirdiğimiz main kod bloğu da aslında bir fonksiyondur. Fonksiyonlar belli bir kod bloğunu birden farklı yerde kullanmak istediğimizde büyük kolaylık sağlarlar. Aynı kod parçasını kullanılmak istenilen yerde tekrar tekrar yazmak yerine onu bir fonksiyon haline getirirsek sadece fonksiyon adını yazarak o kod parçasına ulaşabilir ve kodun satır sayısının gereksiz yere uzamasını engelleyebiliriz. Fonksiyonlar geri dönüşümlü (return) ve geri dönüşsüz (void) fonksiyonlar olarak ikiye ayrılırlar.

Void xxx(void): return kullanılmaz ve fonk()kullanılır Herhangi bir değer girilmeyecek ve herhangi bir değer döndürülmeyecek(kutuya bir şey atılmayacak ve kutudan bir şey alınmayacak)

int xxx(int): return kullanılır ve fonk(a) kullanılır İlk int dönen rakam sayı döndürülecek(return), ikinci int sayı girilecek. Kısaca, sayıgirilecek ve döndürülecek

void xxx(int): return kullanılmaz ve fonk(a) kullanılır. Fonk çağırılınca Sayı girilecek ancak herhangi bir değer döndürülmeyecek (kutuya bir değer girilecek ancak döndürülmeyecek)

int xxx(void): return kullanılır ve fonk() kullanılır. Fonk(a) yazılmaz

FONKSİYON 2 ŞEKİLDE TANIMLANABİLİR 1)..ÖRNEK 1 DE OLDUĞU GİBİ ANA FONKSİYONDAN ÖNCE SADECE İSMİ YAZILIR ANA FONKSİYONDAN SONRA FONKSİYONUN KENDİSİ YAZILIR 2).. ÖRNEK 2 DE OLDUĞU GİBİ ANA FONKSİYONDAN ÖNCE YAZILIR VE BUNDAN SONRA ANA FONKSİYONUNU YAZILMASINA BAŞLANIR.

Örnek : Sayı girilecek ve return =y*y ile döndürülecek

```
#include<stdio.h>
#include<conio.h>
int kare(int);
main()
{
    int a;
    printf("bir sayı giriniz");
    scanf("%d",&a);
    printf("%d",kare(a));
    getch();return 0;
}
int kare (int y)
{
    return y*y;
}
```

döndürme yok (main'den kare'a)
sayı gir (main'den kare'a)

```
include<stdio.h>
#include<conio.h>
void f(int a)
{
    if(a<50)
        printf("kaldınız");
    else
        printf("gectiniz");
}
main()
{
    int not=0;
    printf("not giriniz");
    scanf("%d",&not);
    f(not);
    getch();return 0;
}
```

döndürme yok (f'ten main'e)
sayı girilmi var. (main'den f'e)

```
#include<stdio.h>
#include<conio.h>
int kare (void);
main()
{
    int a;
    printf("sayinin karesi%d",kare());
    getch();return 0;
}
int kare (void)
{
    int x=8;
    return x;
}
```

void olduğu için kare(a) olmaz (main'den kare'a)
return yok (kare'den main'e)

```
#include<stdio.h>
#include<conio.h>
void f(int a)
{
    int i=0,j=0;
    for(i=0;i<a;i++)
    {
        for(j=0;j<=i;j++)
            printf(" ");
        printf("\n");
    }
}
main()
{
    int k=0;
    printf("kac kere yazilsin??");
    scanf("%d",&k);
    f(k);
    f(k);
    getch();return 0;
}
```

döndürme yok (f'ten main'e)
sayı girilmi var. (main'den f'e)

Bir sayı döndürülmediğinden sonuç görünmez

```
#include<stdio.h>
#include<conio.h>
void kare(int);
main()
{
    int a; printf("bir sayı giriniz");
    scanf("%d",&a);
    // printf("sayinin karesi%d",kare(a));
    getch();return 0;
}
void kare (int y)
{
    int x;
    if(y%2==0)
        x=y*y;
    else
        x=(y+1)*(y+1);
    //return x;
}
```

döndürme yok (kare'den main'e)

Örnek::

```
#include<stdio.h>
#include<conio.h>
void toplam(void);
main()
{
    toplam();
    getch();return 0;
}
void toplam(void)
{
    int i,a;
    printf("bir sayı giriniz::");
    scanf("%d",&a);
    for(i=0;i<a;i++)
        printf("nali ozturk");
}
```

return yok (toplam'den main'e)

```
bir sayı giriniz::3
ali ozturk
ali ozturk
ali ozturk
```

Kombinasyon hesabı : Kombinasyon Nedir? Kombinasyon Hesaplama Formülü

n elemanlı bir A kümesinin r elemanlı ($r \leq n$) alt kümelerinin herbirine A kümesinin r li kombinasyonu denir. n elemanlı bir kümenin r li kombinasyonlarının sayısı;

$$C(n,r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Örnek1: kombinasyon hesaplayan program Fonksiyon kullanmasaydık üç tane faktöriyel hesabı yapan kod yazacaktık

```
#include<stdio.h>
#include<conio.h>
int f(int a){
int i=0,carpim=1;
for(i=1;i<=a;i++)
carpim=carpim*i;
return carpim;
}
main(){
int sonuc=0, b=0, c=0;
printf("hangi kombinasyon");
scanf("%d %d",&b,&c);
sonuc=f(b)/(f(b-c)*f(c));
printf("kombinasyon sonucu %d",sonuc);
getch();return 0;}
```

Örnek 2 Girilen sayının karesinin hesaplanması

```
#include<stdio.h>
#include<conio.h>
int kare(int);
main(){
int a;
printf("bir sayi giriniz");
scanf("%d",&a);
printf("%d",kare(a));
getch();return 0;}
int kare (int y){
return y*y;}
```

Örnek 3: faktöriyel hesaplanan fonksiyon

```
#include<stdio.h>
#include<conio.h>
int f(int a){
int i=0,carpim=1;
for(i=1;i<=a;i++)
carpim=carpim*i;
return carpim;
}
main(){
int k=0;
k=f(5);
printf("%d",k); getch();return 0;}
```


Örnek4 ::

```
#include<stdio.h>
#include<conio.h>
int kare(int);
main(){ int a,b; printf("bir sayi giriniz");   scanf("%d",&a);
b=kare(a); //printf("%d",kare(a));
printf("%d",b);getch();return 0;}
int kare (int y){
    return y*y;
}
```

Girilen sayı çift ise karesini alıp yazacak değilse bir ekleyip karesini alacak `int kare(int)` yerine **`void kare (int)`** yazarsak hata verecek çünkü değer girilecek ancak döndürülmeyecek demektir. Ve `return x` yani `x` i döndür hataya sebep olur ayrıca fonksiyon tanımlama ismi ile fonksiyon ismi `int kare(int)` aynı olmalıdır `int kare(int) = int kare(int y)` gibi ayrıca başta fonksiyon tanımlarken noktalı virgül konur `printf` bir değer elde edilemeyeceğinden yazacak bir sonuç bulamaz

Örnek:5 aşağıdaki programda bir sayı döndürülmediğinden `return` kullanmak yanlış olur bir sonuç görülemez sonuç görmek için `void` yerine `int` kullanılması gerekir.

```
#include<stdio.h>
#include<conio.h>
void kare(int);
main(){
int a; printf("bir sayi giriniz");
scanf("%d",&a);
// printf("sayinin karesi%d",kare(a));
getch();return 0;}
void kare (int y){
    int x;
    if(y%2==0)
        x=y*y;
    else
        x=(y+1)*(y+1);
    //return x;
}
```

Örnek 6 Herhangi bir değer girilmeyecekse `void` kullanılır döndürme var değer girme yok Hangi sayıyı girersen gir girmiş olduğunuz sayının karesi 8 mesajı verecek kare fonksiyonu hep 8 değerini verecek `Printf`, `scanf` gereksiz olduğundan çıkartabiliriz

```
#include<stdio.h>
#include<conio.h>
int kare (void);
main(){
int a;
//printf("bir sayi giriniz");
// scanf("%d",&a);
printf("sayinin karesi%d",kare());
getch();return 0;}
int kare (void){
    int x=8;
    return x;}
```

Örnek 7: Girilen 3 sayıdan minimumu bulan fonksiyon

```
#include<stdio.h>
#include<conio.h>
int minimum (int, int, int );
main(){
    int a,b,c;
    printf("uc tam sayi giriniz");
    scanf("\n%d%d%d",&a,&b,&c); printf("sayilarin en kucugu %d",minimum(a,b,c));
    getch();return 0;}
int minimum (int x, int y, int z){
    int min;
    min=x;
    if(y<min)
        min=y;
    if(z<min)
        min=z;
    return min;}
```

Örnek 8 :: Void sayı döndürmeyen fonksiyon değer döndürmeyen sonuç döndürmez Void bilgi değiştirme ekrana bilgi yazma gibi işlerde kullanılır. int sonuç döndürür örneğin faktöriyel sonucunu döndürür.

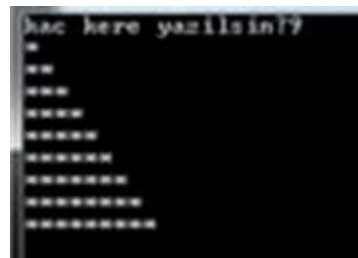
```
#include<stdio.h>
#include<conio.h>
void f(int a){
    if(a<50)
        printf("kaldiniz");
    else
        printf("gectiniz");
    }
main() {
    int not=0;
    printf("not giriniz");
    scanf("\n%d",&not);
    f(not);
    getch();return 0;}

```

Örnek 9: kaç satır istiyorsak her defasında bir arttırarak alt satıra geçip yıldız sembolü yazdıran program

```
#include<stdio.h>
#include<conio.h>
void f(int a){
    int i=0,j=0;
    for(i=0;i<a;i++){
        for(j=0;j<=i;j++){
            printf("*");
            printf("\n");
        }
    }
main() {
    int k=0;
    printf("kac kere yazilsin??");
    scanf("\n%d",&k);
    f(k);
    getch();return 0;}

```



Örnek 10 yukarıdaki örnek ile aynı işi 2 defa yapan program

```
#include<stdio.h>
#include<conio.h>
void f(int a){
    int i=0,j=0;
    for(i=0;i<a;i++)
    {
        for(j=0;j<=i;j++)
            printf("*");
        printf("\n");
    }
}
main() {
    int k=0;
    printf("kac kere yazilsin??");
    scanf("\n%d",&k);
    f(k);
    f(k);
    getch();return 0;}
```

```
kac kere yazilsin??4
**
***
****
*****
**
***
****
*****
```

Örnek 11:: bir sayı girilecek döndürülmeyecek**Kaç defa yazılacağı girilecek döndürme yok**

```
#include<stdio.h>
#include<conio.h>
void toplam(int);
main(){
    int a;
    printf("bir sayi giriniz");
    scanf("%d",&a);
    toplam(a);
    getch();return 0;}
void toplam(int a) {
    int i;
    for(i=0;i<a;i++)
        printf("\nali ozturk");}
```

```
bir sayi giriniz::2
ali ozturk
ali ozturk
```

Örnek 12::

```
#include<stdio.h>
#include<conio.h>
void toplam(void);
main(){
    toplam();
    getch();return 0;}
void toplam(void) {
    int i,a;
    printf("bir sayi giriniz::");
    scanf("%d",&a);
    for(i=0;i<a;i++)
        printf("\nali ozturk");
}
```

```
bir sayi giriniz::3
ali ozturk
ali ozturk
ali ozturk
```

Örnek13

```
#include<stdio.h>
#include<conio.h>
main()
{
    mesaj_yaz();
    getch();return 0;}
void mesaj_yaz(void) {
    puts("Merhaba...");
}
```

ÖRNEK 14

```
/* 08prg04.c: Basit bir kutu çizen fonksiyon
*/
#include <stdio.h>
void kutu_ciz( int satir, int sutun )
{
    int sut;
    for ( ; satir > 0; satir--)
    {
        for (sut = sutun; sut > 0; sut--)
            printf("X");
        printf("\n");
    }
}
int main() {
    kutu_ciz(8,35);
    return 0;}
```

ÇIKTI

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

DİZİLER

Bellekte sürekli bir biçimde bulunan aynı türden nesnelerin oluşturduğu kümeye dizi denir.

Bir diziyi dizi yapan iki temel özellik vardır:

- dizi elemanların bellekte (program çalıştığı sürece) sürekli biçimde bulunması
- dizi elemanların aynı türden değişkenler olması

Bir dizinin bildirim işleminin genel biçimi şöyledir:

Veri_Tipi dizi_adı[eleman_sayısı];

Bir diziyi başlangıç değerleri aşağıdaki gibi kısa formda atanabilir:

```
float kutle[5]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
```

```
int maliyet[3] = { 25, 72, 94 };
```

```
double a[4] = { 10.0, 5.2, 7.5, 0.0};
```

Bir dizinin uzunluğu belirtilmeden de başlangıç değeri atamak mümkündür.

```
int a[] = { 100, 200, 300, 400 };
```

```
float v[] = { 9.8, 11.0, 7.5, 0.0, 12.5};
```

Derleyici bu şekilde bir atama ile karşılaştığında, küme parantezi içindeki eleman sayısını hesaplar ve dizinin o uzunlukta açıldığını varsayar. Yukarıdaki örnekte, a dizisinin 4, v dizisinin 5 elemanlı olduğu varsayılır.

- printf ve scanf fonksiyonları bir dizinin okunması ve yazdırılması için de kullanılır. Örneğin bir A dizisinin aşağıdaki gibi bildirildiğini varsayalım:
- int A[10]; Bu dizinin elemanlarını klavyeden okumak için:

```
for(i=0; i<10; i++)
```

```
scanf("%d",&A[i]);
```

- daha sonra bu değerlerini ekrana yazmak için:

```
for(i=0;i<10;i++)
```

```
printf("%d\n",A[i]);
```

Örneğin, akım verilerini bellekte tutmak için, akım dizisi şöyle tanımlanabilir:

```
float akim[5];
```

Bu dizinin elemanlarına bir değer atama işlemi şöyle yapılabilir:

```
akim [0] = 8.471
```

```
akim [1] = 3.683
```

```
akim [2] = 9.107
```

```
akim [3] = 4.739
```

```
akim [4] = 3.918
```

Bir dizi çok sayıda değişken barındırdığından, bunları birbirinden ayırt etmek için indis adı verilen bir bilgiye ihtiyaç vardır. C Programlama Dili'nde, bir dizi hangi tipte tanımlanmış olursa olsun başlangıç indisi her zaman 0'dır.

```
akim [0] = 8.471
```

```
akim [1] = 3.683
```

Örnek1 dizi tanımlama

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

```
#include<conio.h>
```

```
int main(){
```

```
int a[10] = {25, 18, 20, 0, 29, 5, 4, 8,19,13};
```

```
a[7]= a[7] + a[1];
```

```
printf("%d",a[7]);
```

```
getch();return 0;}
```

Örnek2 dizi tanımlama

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

```
#include<conio.h>
```

```
int main(){
```

```
int ali[3];
```

```
ali[0]=8;
```

```
ali[1]=4;
```

```
ali[2]=5;
```

```
printf("\ndizinin 2.elemanı %d",ali[1]);
```

```
getch();return 0;}
```

Örnek3 dizi tanımlama

```
#include<stdio.h>
#include<conio.h>
int main(){
int i,ali[3];
ali[0]=8;
ali[1]=4;
ali[2]=7;
for(i=0;i<=2;i++)
printf("\n%d.dizinin degeri = %d",i+1,ali[i]);
getch();return 0;}
```

Örnek4 dizi tanımlama

```
#include<stdio.h>
#include<conio.h>
int main(){
int i,ali[3]={8,4,7};
for(i=0;i<=2;i++)
printf("\n%d.dizinin degeri = %d",i+1,ali[i]);
getch();return 0;}
```

Örnek 5 dizinin elemanlarını klavyeden girme

```
#include<stdio.h>
#include<conio.h>
int main(){
int i,ali[3];
printf("dizinin 3 elemaninin giriniz");
for(i=0;i<=2;i++)
scanf("%d",&ali[i]);
for(i=0;i<=2;i++)
printf("\n%d.dizinin degeri = %d",i+1,ali[i]);
getch();return 0;}
```

Örnek6 - Klavyeden girilen n adet sayının toplamını bulan C programı

```
#include<stdio.h>
#include<conio.h>
main(){
int n,toplam=0,i;
puts("Kaç adet sayı gireceksiniz");
scanf("%d",&n);
int dizi[n];
for(i=0;i<n;i++){
printf("%d.sayi:",i+1);
scanf("%d",&dizi[i]);
}
for(i=0;i<n;i++)
toplam=toplam+dizi[i];
printf("dizi elemanları toplamı:%d",toplam);
getch(); return 0;}
```

Örnek 7 elemanlı bir dizinin tek ve çift sayılı elemanlarını ayrı ayrı 2 dizi halinde veren c programı

```
#include<stdio.h>
#include<conio.h>
int main(){

int sayi[10],cift[10],tek[10],i,j=0,k=0;

printf("10 tane sayi giriniz\n");
for(i=0;i<10;i++){
scanf("%d",&sayi[i]);
if (sayi[i]%2==0){
cift[j]=sayi[i];
j++;
}
else{
tek[k]=sayi[i];
k++;
}
}
printf("tum sayilar\n");
for(i=0;i<10;i++)
printf("\n%d",sayi[i]);

printf("\ntek sayilar\n");
for(i=0;i<k;i++)
printf("\n%d",tek[i]);

printf("\ncift sayilar\n");
for(i=0;i<j;i++)
printf("\n%d",cift[i]);
getch(); return 0;}
```

Dizilerde Sıralama

Bazı uygulamalarda bir kısım sayıların büyükten küçüğe ya da küçükten büyüğe sıralanması istenebilir. Bu aslında yazılım uygulamalarının en önemli problemlerinden biridir. Bu problemi çözmek için çeşitli algoritmalar oluşturulmuştur. En basit yöntem ise yer değiştirme yöntemidir. Örnek olarak 5 elemanlı bir diziyi küçükten büyüğe sıralayalım. Sıralamasını yapmak istediğimiz dizi $\text{int dizi}[5] = \{10, 8, 3, 1, 6\}$ olsun. Yer değiştirme işlemi geçici bir değişken kullanılarak yapılır. bu değişkenimizin ismi gecici (temp) olsun.

```
gecici = dizi[i];
dizi[i] = dizi[j];
dizi[j] = gecici;
```

		a[0]	a[1]	a[2]	a[3]	a[4]
i=0	j=0	10	8	3	1	6
i=0	j=1	8	10	3	1	6
	j=2	3	10	8	1	6
	j=3	1	10	8	3	6
	j=4	1	10	8	3	6
i=1	j=2	1	8	10	3	6
	j=3	1	3	10	8	6
	j=4	1	3	10	8	6
i=2	j=3	1	3	8	10	6
	j=4	1	3	6	10	8
i=3	j=4	1	3	6	8	10

Örnek: 8 Basit sıralama ile dizi elemanlarını küçükten büyüğe doğru sıralama.

```
#include<stdio.h>
#include<conio.h>
#define MAX 100
void main() {

    int i, j, n, gecici, dizi[MAX];
    printf("Dizinin eleman sayisini giriniz > ");
    scanf("%d",&n);
    printf("Dizi elemanlarini giriniz > \n\n");
    for(i=0 ; i < n; i++) {
        printf("%3d. eleman: ",i+1);
        scanf("%3d",&dizi[i]);
    }
    printf(" Girilen dizi :");
    for(i=0 ; i < n; i++) {
        printf("%3d",dizi[i]);
    }
    printf("\n");
    for(i=0; i<(n-1); i++) {
        for(j=i+1; j<n; j++)
        {
            if(dizi[i]>dizi[j])
```

```
{
    gecici=dizi[i];
    dizi[i]=dizi[j];
    dizi[j]=gecici;
}
}
}
printf("\n Dizinin Siralı Hali: { ");
for(i=0 ; i < n; i++)
{
    printf("%3d,",dizi[i]);
}
printf(" }");   getch();return 0; }
```

Örnek;9 elemanlı bir dizinin sıralanması

```
#include<stdio.h>
#include<conio.h>
main(){
    int x[5],i,j,gec;
    puts("5 tane sayi gir:");
    for(i=0;i<5;i++) {
        printf("%d.sayi : ",i);
        scanf("%d",&x[i]);
    }
    for(i=0;i<4;i++) {
        for(j=i+1;j<5;j++) {
            if( x[j]<x[i] )
            {
                gec = x[i];
                x[i] = x[j];
                x[j] = gec;
            }
        }
    }
}
```

```
}
}
for(i=0;i<5;i++){
    printf("\n%d",x[i]);
    getch();return 0; }
```

Dizilerde sıralama; en büyük değere sahip olan elemanın dizinin kaçınıcı elemanı olduğunun bulunması:

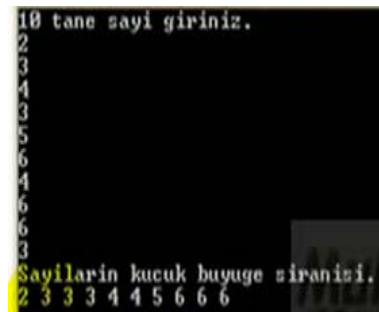
```
for(k=0; k<10; k++){
    if( a[k] > eb){
        eb= a[k];
        i = k;
    }
}
```

Örnek: 10 elemanlı bir dizinin en küçük elemanı ve n büyük elemanını bulan program.

```
#include <stdio.h>
//10 elemanlı bir dizinin en büyük ve en küçük elemanını bulma
main(){
    int a[10] = {100, -250, 400, 125 ,550, 900, 689, 450, 347, 700};
    int k, eb,ek;
    /* ilk eleman en büyük kabul ediliyor */
    eb = a[0];
    ek = a[0];
    for(k=1; k<10; k++)
        { if( a[k]>eb ) eb = a[k];
          if( a[k]<ek ) ek = a[k];}
    printf("En buyuk eleman = %d\n",eb);
    printf("En kucuk eleman = %d\n",ek);
    getch();}
```

Örnek Girilen 11 sayılı diziyi küçükten büyüğe sıralama

```
#include<stdio.h>
#include<conio.h>
int main(){
    int i, hold,tur, a[10];
    printf("10 tane sayi gir:\n");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(tur=0;tur<10;tur++)
        for(i=0;i<9;i++)
            if( a[i]>a[i+1]){ // eğer dizinin bir elemanı bir sonraki elemanından büyükse ikisi yer değiştirsin
                hold=a[i];
                a[i]=a[i+1];
                a[i+1]=hold;
            }
    printf("sayilarin kucukten buyuge siralanisi\n");
    for(i=0;i<10;i++)
        printf("%d",a[i]);
    getch(); return 0; }
```



```
10 tane sayi giriniz.
2
3
3
3
4
4
4
5
6
6
Sayilarin kucuk buyuge siralanisi.
2 3 3 3 4 4 4 5 6 6
```


Örnek Girilen 12 sayılı diziyi büyükten küçüğe sıralama

```
#include<stdio.h>
#include<conio.h>
int main(){
    int i, hold,tur, a[10];
    printf("10 tane sayi gir:\n");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(tur=0;tur<10;tur++)
        for(i=0;i<9;i++)
            if( a[i]<a[i+1]){
                hold=a[i];
                a[i]=a[i+1];
                a[i+1]=hold;
            }

    printf("sayilarin buyukten kucuge siralanisi\n");

    for(i=0;i<10;i++)

        printf("%d",a[i]);

    getch(); return 0; }
```

Örnek 13 elemanlı bir dizinin tek ve çift sayılı elemanlarını ayrı ayrı 2 dizi halinde veren c programı

```
#include<stdio.h>
#include<conio.h>
int main(){

    int sayi[10],cift[10],tek[10],i,j=0,k=0;

    printf("10 tane sayi giriniz\n");
    for(i=0;i<10;i++){
        scanf("%d",&sayi[i]);
        if (sayi[i]%2==0){
            cift[j]=sayi[i];
            j++;
        }
        else{
            tek[k]=sayi[i];
            k++;
        }
    }
    printf("tum sayilar\n");
    for(i=0;i<10;i++)
        printf("\n%d",sayi[i]);

    printf("\ntek sayilar\n");
    for(i=0;i<k;i++)
        printf("\n%d",tek[i]);

    printf("\ncift sayilar\n");
    for(i=0;i<j;i++)
        printf("\n%d",cift[i]);

    getch(); return 0;}
```

örnek 14: *Karakter dizisi (String)*

```

main(){
    int i;
    char ad[][15] = {"Ahmet",
                     "Mehmet","Can"};
    int kilo[]={70,60,52};
    float boy[]={1.70,1.85,1.45};
    puts("\nISIM\tKILO\tBOY");
    for(i=0;i<3;i++){

printf("%s\t%d\t%f\n",ad[i],kilo[i],boy[i]);
    }
}

```

örnek 15: *Beş sayının ortalama hesabı*

```

main(){
    int i,x[5],toplam =0;
    float ort;
    for(i=0;i<5;i++){
        printf("%d. eleman : ",i+1);
        scanf("%d",&x[i]);
        toplam += x[i];
    }
    ort = (float) toplam/5;
    printf("ortalamalari : %f",ort); }

```

Örnek16 : *Dizilerin fonksiyonla kullanma*

```

void yaz(int x[]);
main(){
    int x[10];
    yaz(x);
}
void yaz(int x[]){
    int i;
    for(i=0;i<10;i++){
printf("%d\n",x[i]);
    }
}

```

Örnek17 :: *Dizinin bellekte kapladığı alan*

```

main(){
char dizi1[10];
int dizi2[10];
float dizi3[10];
double dizi4[10];
printf( "%d\n",sizeof(dizi1) );
printf( "%d\n",sizeof(dizi2) );
printf( "%d\n",sizeof(dizi3) );
printf( "%d\n",sizeof(dizi4) ); }

```

Örnek 18'10 lu sistemden 2'li sistem çeviren c programı

```

#include <stdio.h>
#include <conio.h>
//10 ludan 2liye çevirme
main(){
    int ikili[20],sayi,i=0,k;
    printf("Sayı gir:");
    scanf("%d",&sayi);
    while (sayi>0)
    {
        ikili[i]=sayi%2;
        sayi/=2;
        i++;
    }
    for(k=i-1;k>=0;k--)
        printf("%d",ikili[k]);
    getch();return 0;}

```

Örnek19: histogram çizdirme

```

#include<stdio.h>
#include<conio.h>
#define SIZE 10
int main(){
    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
    int i, j;
    printf( " %s%13s%17s\n", "Eleman", "Deger", "Histogram" );
    for( i = 0; i <= SIZE -1; i++ ) {
        printf( "%7d%13d ", i, n[ i ] );
        for( j = 1; j <= n[ i ]; j++ )
            printf( "%c", '*' );
        printf( "\n" );
    }
    getch(); return 0;}

```

MATRİS OLUŞTURMA

3×4 boyutunda 12 elemanlı bir matris aşağıdaki gibi tanımlanabilir;

```
int matris[3][4] = {3, 5, 0, 1, 8, 12, 9, 125, 0, 2, 4, 6};
int matris[3][4] = {{3, 5, 0, 1}, {8, 12, 9, 125}, {0, 2, 4, 6}};
matris[0][0]=3 matris[0][1]=5 matris[0][2]=0 matris[0][3]=1
matris[1][0]=8 matris[1][1]=12 matris[1][2]=9 matris[1][3]=125
.....
```

3	5	0	1
8	12	9	125
0	2	4	6

Örnek 20 matris oluşturma

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int matris[3][4] = {3, 5, 0, 1, 8, 12, 9, 125, 0, 2, 4, 6};
    int i, j;
    for(i=0; i<3; i++) {
        for(j=0; j<4; j++) {
            printf("%4d ", matris[i][j]);
            printf("\n");
        }
        system("pause"); return 0;
    }
}
```

Örnek21 : İstenilen boyutta bir matris oluşturma

```
#include <stdio.h>
#include <conio.h>
int main() {
    int matris[100][100];
    int satir, sutun, i, j;
    printf("Matrisin satir sayisini gir > ");
    scanf("%d",&satir);
    printf("Matrisin sutun sayisini gir > ");
    scanf("%d",&sutun);
    printf("Matrisin degerlerini gir > ");
    for(i=0; i<satir; i++) {
        for(j=0; j<sutun; j++) {
            printf("\n Deger [%u] [%u] --> ", i+1, j+1);
            scanf("%u", &matris[i][j]);
        }
    }
    printf("\nolusturulan matris:\n\n");
    for(i=0; i<satir; i++)
    {
        for(j=0; j<sutun; j++) {
            printf("%3u ", matris[i][j]);
        }
        printf("\n");
    }
    getch();return 0; }
```

örnek 22: 3x3 iki matrisin toplanması

```

#include <stdio.h>
#include <conio.h>
main(){
    int i,j,A[3][3],B[3][3],C[3][3];

    puts("iki matrisin toplami:");

    puts("A matrisinin elemanlarini girin:");

    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            printf("A(%d,%d)=",i+1,j+1);

            scanf("%d",&A[i][j]);

        }
    }

    puts("B matrisinin elemanlarini girin:");

    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            printf("B(%d,%d)=",i+1,j+1);

            scanf("%d",&B[i][j]);

        }
    }

    puts("A+B matrisinin elemanlari:");

    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            C[i][j] = A[i][j] + B[i][j];

            printf("C(%d,%d)=%d\n",i+1,j+1,C[i][j]);

        }
    }

    getch();return 0;}

```

Örnek 23 Determinant 2X2 matrix:

```

#include<stdio.h>
#include<conio.h>
int main(){
    int a[2][2],i,j;
    long determinant;
    printf("Enter the 4 elements of matrix: ");
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
            scanf("%d",&a[i][j]);
    printf("\nThe matrix is\n");
    for(i=0;i<2;i++){
        printf("\n");
        for(j=0;j<2;j++)
            printf("%d\t",a[i][j]);
    }
    determinant = a[0][0]*a[1][1] - a[1][0]*a[0][1];
    printf("\nDeterminant of 2X2 matrix: %ld",determinant);
    getch(); return 0;}

```

MATRİS ÇAPIMI

```

for (int i = 1;i<=m ; i++){
    for (int j = 1;i<=n ;j++){
        for(int k = 1;k<=p;k++){
            c[i][j] = a[i][k]*b[k][j] + c[i][j];
        }
    }
}

```

Örnek 22: 2 matrisi çarpan program

```
#include <stdio.h>
#include <conio.h>
main() {
    int a[3][2],b[2][3],c[3][3] = { 0 },i,j,k;
    printf ("Birinci Matris: n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 2; j++) {
            printf (" [%d,%d]: ",i+1,j+1);
            scanf ("%d", &a[i][j]);
        }
    }
    printf ("Ikinci Matris: n");
    for (j = 0; j < 2; j++) {
        for (k = 0; k < 3; k++) {
            printf (" [%d,%d]: ",j+1,k+1);
            scanf ("%d", &b[j][k]);
        }
    }
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            for (k = 0; k < 2; k++)
                c[i][j] += a[i][k] * b[k][j];
        }
        printf("n");
    }
    printf ("Sonuc:n");
    for (i = 0; i < 3; i++) {
        for (k = 0; k < 3; k++)
            printf ("t%d",c[i][k]);
        printf ("n");
    }
    getch(); return 0; }
```

Katarlar (Stringler)

katar = karakter topluluğu = karakter dizisi = sözce = sicim

Katar Bildirimi

```
char s[5]={ 'D', 'U', 'Z', 'C', 'E', '\0' };
char s[5]="DUZCE";
char s[] = "Ankara";    /* 6 elemanlı */
```

Gösterici Bildirimi

```
char *s = "Ankara";    /* 6 elemanlı */

char *s;
s = "Ankara";
```

Ancak

```
char s[6];
s = "Ankara";
```

şeklindeki bir atama geçersizdir. Çünkü bu şekilde yapılan bildirimde s bir değişken değil dizidir. Elemanları katar olan diziler tanımlamak mümkündür. Örneğin en uzun 7 karakter olan 5 farklı isim bir çatı altında şöyle toplanabilir:

```
char isim[5][8] = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

yada

```
char isim[][8] = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

yada

```
char *isim[5] = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

Görüldüğü gibi, bu tip tanımlamalarda birinci boyut (satır) dizinin eleman sayısını, ikinci boyut (sütun) her bir elemanın sahip olabileceği maksimum karakter sayısını gösterir.

Katar ifadelerinde doğrudan çift tırnak " veya ters bölü \ karakterleri kullanılamaz. Bu durumda katar ifadeleri içerisinde

12.3 Katarlar Üzerinde İşlem Yapan Standart G/Ç Fonksiyonları

`printf()` ve `scanf()` fonksiyonları diğer tiplerde olduğu gibi formatlı okuma/yazma amaçlı kullanılır. Katar formatı `%s` dir. Örneğin:

```
char str[20];
...
scanf("%s",str);
printf("%s\n",str);
```

satırları ile klavyeden okunan katarın ilk 20 karakteri ekrana yazdırılabilir. Burada `printf()` fonksiyonu:

```
printf(str);
```

şeklinde de kullanılabilir. Bu durumda, katar ekrana yazdırılır fakat imlec (cursor) bir alt satıra geçmez.

`gets()` fonksiyonu klavyeden karakter dizisi almakta kullanılan bir C fonksiyonudur. Bu fonksiyon, klavyeden girilen karakterleri diziye yerleştirdikten sonra dizinin sonuna otomatik olarak `NULL ('\0')` karakterini ekler.

```
char str[20];
...
gets(str);
```

NOT :::::

gets() fonksiyonunu kullanmak biraz tehlikeli olabilir.

Çünkü, gets() ile okuma yapılırken katarın büyüklüğünü dikkate alınmaz. Örneğin:

```
char s[10];
gets(s);
```

şeklindeki okuma işleminde s en fazla 10 karakter saklayabilirken, gets() ile 100 karakter girilirse, derleyici bütün karakterleri saklamaya çalışır. Bu durumda, program sağlıklı çalışmaz ve hata verir. Bu yüzden bazı derleyiciler, gets() kullanıldığında aşağıdaki gibi bir uyarı verir.

warning: the 'gets' function is dangerous and should not be used.

Sonuç olarak, scanf() fonksiyonunu kullanmanız tavsiye edilir.

puts() fonksiyonu bir karakter dizisini ekrana yazdırmak için kullanılır. Bu fonksiyon diziyi ekrana yazdırdıktan sonra imleci (cursor) bir sonraki satıra geçirir.

```
#include <stdio.h>
#include <conio.h>
int main(){
char *str = "Hangi cilgin bana zincir vuracakmis sasirim";
puts(str);
getch(); return 0;}
```

puts(str) ile printf("%s\n", str) işlevsel olarak birbirine eşdeğerdir.

/* 01. Bir katarın farklı yöntemlerle ekrana yazılması */

```
#include <stdio.h>
#include <conio.h>
int main(){

char dizi[7] = {'S', 'e', 'l', 'a', 'm', '!', '\0'};
int i;
/* Herbir karakteri ayrı ayrı alt alta yaz */
printf("Dizi elemanlari:\n");
for (i=0; i<7; i++)
    printf("dizi[%d] icerigi: %c\n", i, dizi[i]);
printf("\n");

/* 1. yöntem: her elemanı yanyana yaz */
printf("Butun dizi (1.yontem): ");
for (i=0; i<7; i++)
    printf("%c", dizi[i]);

/* 2. Yöntem: bütün diziyi yaz */
printf("\nButun dizi (2.yontem): ");
printf("%s\n", dizi);
printf("\n");
getch();return 0;}
```

PROF. DR. ALİ ÖZTÜRK

ÇIKTI

Dizi elemanlari:
 dizi[0] icerigi: S
 dizi[1] icerigi: e
 dizi[2] icerigi: l
 dizi[3] icerigi: a
 dizi[4] icerigi: m
 dizi[5] icerigi: !
 dizi[6] icerigi:

Butun dizi (1.yontem): Selam!
 Butun dizi (2.yontem): Selam!

/* 02.c: Bir stringin içindeki 'm' karakterlerinin sayısı hesaplar */

#include <stdio.h>

#include <conio.h>

int main(){

char str[20];

int i,sayac=0;

printf("Bir string girin: ");

gets(str);

for(i=0; str[i] != '\0'; i++)

if(str[i] == 'm') sayac++;

printf("'m' karakteri sayisi = %d\n",sayac);

getch(); return 0;}

ÇIKTI

Bir katar girin: **marmara**

'm' karakteri sayisi = 2

döngü şöyle de yazılabilirdi:

for(i=0; str[i]; i++)

if(str[i] == 'm') sayac++;

Buradaki işleme str[i], NULL karakterinden farklı olduğu sürece döngü devam ettirilmiştir.

/* 03.c: Bir elemanları katar olan karakter dizisini yazdırma */

#include <stdio.h>

#include <conio.h>

int main(){

**char *gun[7] = { "Pazartesi", "Sali", "Carsamba",
 "Persembe", "Cuma", "Cumartesi", "Pazar" };**

int i;

for(i=0; i<7; i++)

printf("%d. %s\n",i+1,gun[i]);

getch(); return 0;}

ÇIKTI

1. Pazartesi
 2. Sali
 3. Carsamba
 4. Persembe
 5. Cuma
 6. Cumartesi
 7. Pazar

Örnekler;

1)

```
#include<stdio.h>
#include<conio.h>
int main( void ){
    char isim[30];
    printf( "İsim giriniz> ");
    scanf( "%s", isim );
    printf( "Girdiğiniz isim: %s\n", isim );
    getch(); return 0; }
```

```
#include<stdio.h>
#include<conio.h>
int main( void ){
    char cumle[40];
    printf( "Cümle giriniz> ");
    gets( cumle );
    printf( "Girdiğiniz cümle:\n" );
    puts( cumle );
    getch();return 0;}
```

2)

```
#include<stdio.h>
#include<conio.h>
int main( void ) {
    char isim[30];
    int i;
    printf( "İsim giriniz> ");
    scanf( "%s", isim );

    printf( "Girdiğiniz isim: ");
    for( i = 0; isim[i]!='\0'; i++ )
        printf( "%c", isim[i] );
    printf("\n");

    getch(); return 0;}
```

5)

```
#include<stdio.h>
#include<conio.h>
int main( void ){

    char isim[] = "Ali";
    char soyad[5] = "ozturk";
    printf( "%s %s\n", isim, soyad );

    getch();return 0;}
```

3)

```
#include<stdio.h>
#include<conio.h>
int main( void ){
    char isim[25], soyad[30];
    printf( "Ad ve soyad giriniz> ");
    scanf( "%s%s", isim, soyad );
    printf( "Sayın %s %s, hoş geldiniz!\n",
    isim, soyad );
    getch(),return 0;}
```

6)

```
#include<stdio.h>
#include<conio.h>
int main( void ){
    char isim[] = { 'A', 'L', 'T', '\0' };
    char soyad[7] = { 'O', 'Z', 'T', 'U', 'R', 'K', '\0' };
    printf( "%s %s\n", isim, soyad );
    getch(); return 0;}
```

4) scanf ile cumle giremeyiz boşuk Kabul etmez
merhaba sınıf darken sadece merhaba yazılır bunun
için gets puts ikilisi kullanılır

string.h kütüphanesine ait, bazı katar fonksiyonları

Fonksiyon	Açıklama
<code>int strcmp(char *str1, char *str2);</code>	str1 ve str2 yi karşılaştırır. Eşitse 0, str1 büyükse 0'dan büyük bir değer aksi halde 0'dan küçük bir değer gönderir.
<code>char *strcpy(char *str1, char *str2);</code>	str2 yi str1 e kopyalar
<code>char *strcat(char *str1, char *str2);</code>	str2 yi str1 e ekler
<code>char *strrev(str);</code>	str yi ters çevirir (NULL karakteri hariç)
<code>int strlen(str);</code>	str nin kaç karakterden oluştuğunu hesaplar
<code>char *strchr(char *str, char kr);</code>	kr karakterinin str içindeki (baştan itibaren) ilk karşılaştığı yeri verir
<code>char *strstr(char *str1, char *str2);</code>	str2 katarının str1 içindeki (baştan itibaren) ilk karşılaştığı yeri verir
<code>char *strlwr(char *str);</code>	str nin bütün karakterini küçük harfe çevirir
<code>char *strupr(char *str);</code>	str nin bütün karakterini büyük harfe çevirir

Program 04 : strcmp fonksiyonunun kullanımı

```
/* 12prg04.c: Basit bir şifre programı.*/
```

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
int main() {

    char sifre[8];
    int sonuc, hak=3;

    while( hak-- > 0 )
    {
        printf("Sifre : ");
        gets(sifre);    /* şifreyi al */

        sonuc = strcmp(sifre, "elma%xj4");

        if( sonuc==0 ){    /* şifre kontrol */
            puts("sifre dogru");
            break;
        }
        else
            puts("sifre yanlis");
    }
    getch(); return 0;}
```

```
ÇIKTI
Şi fre : admi n
```

```

si fre yanli s
Si fre : root
si fre yanli s
Si fre : el ma%xj 4
si fre dogru

```

```
program/* 05.c: Bir katarı diğetine kopyalama */
```

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
int main(){
    char str1[] = "Deneme";
    char str2[15], str3[15];
    int i;

    /* strcpy kullanarak kopyalama */
    strcpy(str2, str1);

    /* strcpy kullanmadan kopyalama */
    for(i=0; str1[i]; i++)
        str3[i] = str1[i];
    str3[i] = '\0'; /* sonlandırıcı ekle */

    /* sonuçlar ekrana */
    printf("str1 : %s\n", str1);
    printf("str2 : %s\n", str2);
    printf("str3 : %s\n", str3);
    getch(9; return 0;};

```

ÇIKTI

```

str1 : Deneme
str2 : Deneme
str3 : Deneme

```

Program06 : strcat *fonksiyonunun kullanımı*

```
/* 12prg06.c: Bir katarı diğetine ekler */
```

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
int main(){

    char mesaj[20] = "Selam "; /* 1. katar */
    char isim[10]; /* 2. katar */

    printf("Adiniz ? : ");
    scanf("%s", isim);

    /* ekle */
    strcat(mesaj, isim);

    printf("%s\n", mesaj);
    getch(9; return 0;};

```

ÇIKTI

```

Adi ni z ? : Mert
Sel am Mert

```

Program07 : strlen fonksiyonunun kullanımı

```

/* 12prg07.c: Bir karakter dizisinin uzunluğunu bulur */
#include <stdio.h>
#include <string.h>
#include <conio.h>
int main() {
    char s[20];
    int k = 0;

    printf("Bir seyler yazın : ");
    scanf("%s", s);

    /* sonlandırıcı karaktere kadar */
    while( s[k] != '\0' )
        k++;
    puts("Dizinin uzunluğu");
    printf("strlen kullanarak = %d\n", strlen(s));
    printf("strlen kullanmadan = %d\n", k);
    getch(9; return 0;};

```

ÇIKTI

```

Bir seyler yazın : deneme
stringi
Dizinin uzunluğu
strlen kullanarak = 14
strlen kullanmadan = 14

```

Program : Isim sırlama

```

/*08.c Kabarcık Sıralama Algoritması
ile isimleri alfabetik sırayla listeler */
#include <stdio.h>
#include <string.h>
#include <conio.h>
int main() {
#define n 5
int main()
{
    char isim[n][8] = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
    char gecici[8];
    int i, j, k;

    puts("Once:\n-----");
    for(i=0; i<n; i++)
        printf("%s\n", isim[i]);

    /* sırala */
    for(k=0; k<n-1; k++)
    for(j=0; j<n-1; j++)
        if( strcmp(isim[j], isim[j+1]) > 0 ) /* isim[j]>isim[j+1] ? */
        {
            strcpy(gecici, isim[j]);
            strcpy(isim[j], isim[j+1]);
            strcpy(isim[j+1], gecici);
        }

    puts("\nSonra:\n-----");
    for(i=0; i<n; i++)
        printf("%s\n", isim[i]);

    getch(9; return 0;};

```

ÇIKTI

Önce:

Semra
Mustafa
Ceyhun
Aslı
Leyla

Sonra:

Aslı
Ceyhun
Leyla
Mustafa
Semra

Katarların Fonksiyonlarda Kullanılması

Katarların fonksiyonlara parametre olarak geçirilmesi durumuna sıklıkla rastlanır. Gerçekte fonksiyona parametre olarak aktarılan karakter dizisini gösteren bir adrestir. Bu yüzden karakter dizileri fonksiyonlara çoğunlukla gösterici tipinde geçirilir.

Aşağıdaki iki örnekte yazılan `struzn` ve `strcev` fonksiyonları sırasıyla `strlen` ve `strrev` fonksiyonların dengi niteliğindedir. Burada kullanılan benzer mantıkla, `string.h` kütüphanesindeki birçok fonksiyon yazılabilir. İnceleyiniz.

Program 0.9: `strlen` dengi bir fonksiyon: `struzn`

`/* 12prg09.c: Bir katarın uzunluğunu bulan strlen dengi bir fonksiyon */`

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

int struzn(char *);
int main() {
    char *s;

    printf("Bir katar girin: ");
    gets(s);
    printf("Uzunlugu (struzn) : %d\n", struzn(s));
    printf("Uzunlugu (strlen) : %d\n", strlen(s));
    return 0; }
/* bir karakter dizisinin uzunluğunu hesaplar */
int struzn(char *str)
{
    int n = 0;
    while(str[n])
        n++;
    return n; }
```

ÇIKTI

Bir katar girin: Programlama
Uzunlugu (struzn) : 11
Uzunlugu (strlen) : 11

Program : strrev dengi bir fonksiyon: strcev

```
/* 12prgl0.c: Bir katarın tersini veren bir fonksiyon */
#include <stdio.h>
#include <string.h>

char *strcev(char *);

int main(){
    char s[50];

    printf("Bir katar girin: ");
    scanf("%s",s);

    printf("Katar, s : %s\n",s);
    printf("Tersi, strcev(s) : %s\n",strcev(s));

    return 0;
}

/* str katarını ters-yüz eder */
char *strcev(char *str)
{
    int i,n;
    char gecici;

    n = strlen(str);

    for(i=0; i<n/2; i++)
    {
        gecici      = str[i];
        str[i]      = str[n-i-1];
        str[n-i-1] = gecici;
    }

    return str; }/* geri dönüş değeri bir gösterici */
```

ÇIKTI

```
Bir katar girin: Programlama
Katar, s : Programlama
Tersi, strcev(s) : amal margorP
```

ÖRNEKLER

```
#include<stdio.h>
#include<conio.h>
int main( void ){
    char isim[30];
    printf( "İsim giriniz> ");
    scanf( "%s", isim );
    printf( "Girdiğiniz isim: %s\n", isim );
    getch();return 0;}
```

Örneğimizde 30 karakterlik bir karakter dizisi tanımlayarak işe başladık. Bunun anlamı girdileri saklayacağımız '*isim*' katarının 30 karakter boyutunda olacağıdır. Ancak bu kata en fazla 29 karakterlik bir kelime atanabilir. Çünkü katarlarda, kelime bitiminden sonra en az bir hücre boş bırakılmalıdır. Bu hücre '*Boş Karakter*' (*NULL Character*) tutmak içindir. Boş karakter "**\0**" şeklinde ifade edilir. C programlama dilinde, kelimelerin bittiğini boş karakterlerle anlarız. Herhangi bir katarı boş karakterle sonlandırmaya, '*null-terminated*' denmektedir.

Bu arada katarlara değer atarken ya da katarlardan değer okurken, sadece katar adını yazmamızın yettiğini farketmişsinizdir. Yani scanf() fonksiyonu içersine & işareti koymamız gerekmiyor. Çünkü scanf(), katarın ilk adresinden başlayarak aşağıya doğru harfleri tek tek ataması gerektiğini biliyor. (Aslında biliyor demek yerine, fonksiyonun o şekilde yazıldığını söylememiz daha doğru olur.)

```
#include<stdio.h>
#include<conio.h>
int main( void ) {
    char isim[30];
    int i;
    printf( "İsim giriniz> ");
    scanf( "%s", isim );

    printf( "Girdiğiniz isim: ");
    for( i = 0; isim[i]!='\0'; i++ )
        printf( "%c", isim[i] );
    printf("\n");

    getch(); return 0;}
```

Daha önce tek bir printf() fonksiyonuyla bütün katarı yazdırabilirken, bu sefer katar elemanlarını tek tek, karakter karakter yazdırmayı tercih ettik. Çıkan sonuç aynı olacaktır fakat gidiş yolu biraz farklılaştı. Özellikle *for* döngüsü içersinde bulunan "*isim[i]!='\0'*" koşuluna dikkat etmek gerekiyor. İsteseydik, "*i < 30*" yazar ve katarın bütün hücrelerini

birer birer yazdırabilirdik. Fakat bu mantıklı değil! 30 karakterlik bir dizi olsa bile, kullanıcı 10 harften oluşan bir isim girebilir. Dolayısıyla kalan 20 karakteri yazdırmaya gerek yoktur. Kelimenin nerede sonlandığını belirlemek için "*isim[i] != '\0'*" koşulunu kullanıyoruz. Bunun anlamı; *isim* katarının elemanları, "\0" yani boş karaktere (NULL Character) eşit olmadığı sürece yazdırmaya devam edilmesidir. Ne zaman ki kelime biter, sıradaki elemanın değeri "\0" olur; işte o vakit döngüyü sonlandırmamız gerektiğini biliriz.

Yukardaki örneğimize birden çok kelime girdiyseniz, sadece ilk kelimenin alındığını farketmişsinizdir. Yani "*Bugün hava çok güzel.*" şeklinde bir cümle girdiğiniz zaman, kataşa sadece "*Bugün*" kelimesi atanır. Eğer aynı anda birden fazla kelime almak istiyorsanız, ayrı ayrı belirtilmesi gerekir.

```
#include<stdio.h>
int main( void ){
    char isim[25], soyad[30];
    printf( "Ad ve soyad giriniz> " );
    scanf( "%s%s", isim, soyad );
    printf( "Sayın %s %s, hoş geldiniz!\n", isim, soyad );
    return 0;}
```

gets() ve puts() Fonksiyonları

Gördüğünüz gibi aynı anda iki farklı kelime alıp, ikisini birden yazdırdık. Fakat scanf() fonksiyonu "*Bugün hava çok güzel.*" cümlesini tek bir kataşa alıp, atamak için hâlen yetersizdir. Çünkü boşluk gördüğü noktada, veriyi almayı keser ve sadece "*Bugün*" kelimesinin atamasını yapar. Boşluk içeren bu tarz cümleler için puts() ve gets() fonksiyonları kullanılmaktadır. Aşağıdaki örnek program, 40 harf geçmeyecek her cümleyi kabul edecektir:

```
#include<stdio.h>
int main( void ){
    char cumle[40];
    printf( "Cümle giriniz> " );
    gets( cumle );
    printf( "Girdiğiniz cümle:\n" );
    puts( cumle );
    return 0;}
```

gets() isminden anlayacağınız (*get string*) gibi kataşa değer atamak için kullanılır. puts() (*put string*) ise, bir katarın içeriğini ekrana yazdırmaya yarar. gets() atayacağı değerini ayırmayı yapabilmek için '\n' aramaktadır. Yani klavyeden Enter'a basılana kadar girilen her şeyi, tek bir kataşa atayacaktır. puts() fonksiyonuysa, printf() ile benzer çalışır. Boş karakter (NULL Character) yani '\0' ulaşana kadar katarı yazdırır; printf() fonksiyonundan farklı olarak sonuna '\n' koyarak bir alt satıra geçer. Oldukça açık ve basit kullanımlara sahip olduklarından, kendiniz de başka örnekler deneyebilirsiniz.

Katarlara İlk Değer Atama

Bir katar tanımı yaptığınız anda, katarın bütün elemanları otomatik olarak '\0' ile doldurulur. Yani katarın bütün elemanlarına boş karakter (NULL Character) atanır. Dilerseniz, katarı yaratırken içine farklı değerler atayabilirsiniz. Katarlarda ilk değer ataması iki şekilde yapılır.

Birinci yöntemle değer ataması yapacaksanız, istediğiniz kelimeyi bir bütün olarak yazarsınız:

```
#include<stdio.h>
int main( void )
{
    // Her iki katarada ilk deger
    // atamasi yapiliyor. Ancak
    // isim katarinda, boyut
    // belirtilmezken, soyad katarinda
    // boyutu ayrica belirtiyoruz.
    char isim[] = "Ali";
    char soyad[5] = "ozturk";
    printf( "%s %s\n", isim, soyad );

    return 0;
}
```

İkinci yöntemdeyse, kelime bütün olarak yazılmaz. Bunun yerine harf harf yazılır ve sonlandırmak için en sonuna boş karakter (NULL) eklenir:

```
#include<stdio.h>
int main( void )
{
    char isim[] = { 'A', 'L', 'I', '\0' };
    char soyad[7] = { 'O', 'Z', 'T', 'U', 'R', 'K', '\0' };
    printf( "%s %s\n", isim, soyad );
    return 0;
}
```

Ben ilk değer ataması yapacağım durumlarda, ilk yolu tercih ediyorum. İkinci yöntem, daha uzun ve zahmeti...

Biçimlendirilmiş (Formatlı) Gösterim

Daha önce float tipindeki bir sayının, noktadan sonra iki basamağını göstermek türünden şeyler yapmıştık. Örneğin printf() fonksiyonu içerisinde, sayıyı %.2f şeklinde ifade ederseniz, sayının virgülden sonra sadece iki basamağı gösterilir. Yada %5d yazarak tam sayıları gösterdiğiniz bir durumda, sayı tek bir rakamdan dahi oluşsa, onun için 5 rakamlık gösterim yeri ayrılır. Aynı şekilde biçimlendirilmiş (formatlı) gösterim, katarlarda da yapılmaktadır.

Katarları biçimlendirilmiş şekilde göstermeyi, örnek üzerinden anlatmak daha uygun olacaktır:

```
#include<stdio.h>
int main( void )
{
    char cumle[20] = "Denemeler";

    // Cumleyi aynen yazar:
    printf( "%s\n", cumle );

    // 20 karakterlik alan ayirir
    // ve en saga dayali sekilde yazar.
    printf( "%20s\n", cumle );

    // 20 karakterlik alan ayirir
    // ve en saga dayali sekilde,
    // katarin ilk bes kelimesini
    // yazar
    printf( "%20.5s\n", cumle );

    // 5 karakterlik alan ayirir
    // ve en saga dayali sekilde yazar.
    // Eger girilen kelime 5 karakterden
    // buyukse, kelimenin hepsi yazilir.
    printf( "%5s\n", cumle );

    // 20 karakterlik alan ayirir
    // ve sola dayali sekilde yazar.
    // Sola dayali yazilmasi icin
    // yuzde isaretinden sonra, -
    // (eksi) isareti konulur.
    printf( "%-20s\n", cumle );

    return 0;
}
```

Örneğimizde bulunan formatlama biçimlerini gözden geçirsek:

- %20s, ekranda 20 karakter alan ayrılacağı anlamına gelir. Katar, en sağa dayanır ve "Denemeler" yazılır.
- %.5s olursa 5 karakterlik boşluk ayrılır. Yüzde işaretinden sonra nokta olduğu için katarın sadece ilk beş harfi yazdırılır. Yani sonuç "Denem" olacaktır. %20.5s yazıldığında, 20 karakterlik boşluk ayrılması istenmiş ancak katarın sadece ilk 5 harfi bu boşluklara yazılmıştır.
- %5s kullanırsanız, yine 5 karakterlik boşluk ayrılacaktır. Ancak yüzdeden sonra nokta olmadığı için, katarın hepsi yazılır. Belirtilen boyutu aşan durumlarda, eğer noktayla

sınır konmamışsa, katar tamamen gösterilir. Dolayısıyla çıktı, "Denemeler" şeklinde olacaktır.

- Anlattıklarımızın hepsi, sağa dayalı şekilde çıktı üretir. Eğer sola dayalı bir çıktı isterseniz, yüzde işaretinden sonra '-' (eksi) işareti koymanız gerekir. Örneğin %-20.5s şeklinde bir format belirlerseniz, 20 karakterlik boşluk ayarlandıktan sonra, sola dayalı olarak katarın ilk 5 harfi yazdırılacaktır. İmleç (cursor), sağ yönde 20 karakter sonrasına düşecektir.

Standart Katar Fonksiyonları

Katarlarla daha kolay çalışabilmek için, bazı hazır kütüphane fonksiyonlarından bahsedeceğiz. Bu fonksiyonlar, string kütüphanesinde bulunuyor. Bu yüzden, programınızın başına, `#include<string.h>` eklemeniz gerekiyor.

* `strlen()` fonksiyonuyla katar boyutu bulma

Dizi boyutuyla, katar uzunluğunun farklı şeyler olduğundan bahsetmiştik. Dizi boyutu, 40 karakter olacak şekilde ayarlanmışken, dizi içinde sadece 7 karakterlik "Merhaba" kelimesi tutulabilir. Bu durumda, dizi boyutu 40 olmasına rağmen, katar boyutu yalnızca 7'dir. Katarların boyutunu saptamak için, boş karakter (NULL Character) işaretinin yani "\0" simgesinin konumuna bakılır. Her seferinde arama yapmanıza gerek kalmaması diye `strlen()` fonksiyonu geliştirilmiştir. `strlen()` kendisine argüman olarak gönderilen bir katarın boyutunu geri döndürür. Aşağıdaki gibi kullanılmaktadır:

```
#include<stdio.h>
#include<string.h>
int main( void )
{
    printf( "Katar Uzunluğu: %d\n", strlen("Merhaba") );
    return 0;
}
```

* `strcpy()` ve `strncpy()` ile katar kopyalama

Bir katarı, bir başka katara kopyalamak için `strcpy()` fonksiyonunu kullanırız. Katarlar aynı boyutta olmak zorunda değildir. Ancak kopya olacak katar, kendisine gelecek kelimeyi alacak boyuta sahip olmalıdır. Fonksiyon prototipi aşağıdaki gibidir, geriye pointer döner.

```
char *strcpy( char[ ], char[ ] );
```

`strcpy()` fonksiyonunu bir örnekle görelim:

```
#include<stdio.h>
#include<string.h>
int main( void )
{
```

```

char kaynak[40]="Merhaba Dünya";
char kopya[30] = "";
strcpy( kopya, kaynak );
printf( "%s\n", kopya );

return 0;
}

```

strcpy() fonksiyonu, yine kopyalamak içindir. Fakat emsalinden farklı olarak, kaç karakterin kopyalanacağı belirtilir. Protopi aşağıda verilmiştir:

```

char *strncpy( char[ ], char[ ], int );

```

Yukardaki örneği strncpy() fonksiyonuyla tekrar edelim:

```

#include<stdio.h>
#include<string.h>
int main( void )
{
    char kaynak[40]="Merhaba Dünya";
    char kopya[30] = "";
    strncpy( kopya, kaynak, 9 );
    printf( "%s\n", kopya );

    return 0;
}

```

Yukardaki programı çalıştırırsanız, kopya isimli katarla sadece 9 karakterin aktarıldığını ve ekrana yazdırılan yazının "*Merhaba D*" olduğunu görebilirsiniz.

* strcmp() ve strncmp() ile katar karşılaştırma

strcmp() fonksiyonu, kendisine verilen iki katarı birbiriyle karşılaştırır. Katarlar birbirine eşitse, geriye 0 döner. Eğer ilk katar alfabetik olarak ikinciden büyükse, geriye pozitif değer döndürür. Şayet alfabetik sırada ikinci katar birinciden büyükse, geriye negatif değer dönmektedir. Bu dediklerimizi, daha iyi anlaşılması için bir tabloya dönüştürelim:

Dönen Değer	Açıklama
< 0	Katar1, Katar2'den küçüktür.
0	Katar1 ve Katar2 birbirine eşittir.
> 0	Katar1, Katar2'den büyüktür.

strncmp() için de aynı kurallar geçerlidir. Tek fark, karşılatırlacak karakter sayısını girmemizdir. strcmp() fonksiyonunda iki katar, *null* karakter işareti çıkana kadar karşılaştırılır. Fakat strncmp() fonksiyonunda, başlangıçtan itibaren kaç karakterin karşılatırlacağına siz karar verirsiniz.

Her iki fonksiyonu da kapsayan aşağıdaki örneği inceleyelim:

```
#include<stdio.h>
#include<string.h>
int main( void )
{
    int sonuc;
    char ilk_katar[40]="Maymun";
    char ikinci_katar[40]="Maytap";
    sonuc = strcmp( ilk_katar, ikinci_katar );
    printf( "%d\n", sonuc );
    sonuc = strncmp( ilk_katar, ikinci_katar, 3 );
    printf( "%d\n", sonuc );

    return 0;}

```

İlk önce çağrılan *strcmp()*, null karakterini görene kadar bütün karakterleri karşılaştıracak ve geriye negatif bir değer döndürecektir. Çünkü "*Maymun*" kelimesi alfabede "*Maytap*" kelimesinden önce gelir; dolayısıyla küçüktür. Fakat ikinci olarak çağırdığımız *strncmp()* geriye 0 değeri verecektir. Her iki katarın ilk üç harfi aynıdır ve fonksiyonda sadece ilk üç harfin karşılaştırılmasını istediğimizi belirttik. Dolayısıyla karşılaştırmanın sonucunda 0 döndürülmesi normaldir.

* *strcat()* ve *strncat()* ile katar birleştirme

strcat() ve *strncat()* fonksiyonları, bir katarı bir başka katarla birleştirmeye yarar. Fonksiyon adlarında bulunan *cat*, İngilizce bir kelime olan ve birleştirme anlamına gelen '*concatenate*'den gelmiştir. *strcat()* kendisine verilen katarları tamamen birleştirirken, *strncat()* belirli bir eleman sayısına kadar birleştirir. *strcat* ile ilgili basit bir örnek yapalım.

```
#include<stdio.h>
#include<string.h>
int main( void )
{
    char ad[30], soyad[20];
    char isim_soyad[50];
    printf( "Ad ve soyadınızı giriniz> " );
    scanf( "%s%s", ad, soyad );
    // isim_soyad <-- ad
    strcat( isim_soyad, ad );
    // isim_soyad <-- ad + " "
    strcat( isim_soyad, " " );
    // isim_soyad <-- ad + " " + soyad
    strcat( isim_soyad, soyad );
    printf( "Tam İsim: %s\n", isim_soyad );
    return 0;}

```

Dilerseniz, *strncat()* fonksiyonunu da siz deneyebilirsiniz.

* *strstr()* fonksiyonuyla katar içi arama yapma

Bir katar içinde, bir başka katarı aradığınız durumlarda, *strstr()* fonksiyonu yardımınıza yetişir. *strstr()* fonksiyonu, bir katar içinde aradığınız bir katarı bulduğu takdirde bunun bellekteki adresini geriye döndürür. Yani dönen değer çeşidi bir pointer'dır. Eğer herhangi bir eşleşme olmazsa geriye bir sonuç dönmez ve pointer *null* olarak kalır. Elbette insanlar için hafıza adreslerinin veya pointer değerlerinin pek bir anlamı olmuyor. Bir katar içinde arama yapıyorsanız, aradığınız yapının katarın neresinde olduğunu tespit etmek için aşağıdaki kodu kullanabilirsiniz:

```
/* strstr( ) fonksiyon ornegi */
#include<stdio.h>
#include<string.h>
int main( void )
{
    char adres[] = "Esentepe Caddesi Mecidiyekoy Istanbul";
    char *ptr;
    // 'adres' katarı icinde, 'koy' kelimesini
    // arıyoruz. Bu amacla strstr( ) fonksiyonunu
    // kullanıyoruz. Fonksiyon büyük-kucuk harf
    // duyarlıdır. Eger birden fazla eslesme varsa,
    // ilk adres degeri doner. Hic eslesme olmazsa,
    // pointer degeri NULL olur.
    ptr = strstr( adres, "koy" );
    if( ptr != NULL )
        printf( "Başlangıç notkası: %d\n", ptr - adres );
    else
        printf( "Eşleşme bulunamadı.\n" );
    return 0;
}
```

* *strchr()* ve *strrchr()* fonksiyonları

strchr() ve *strrchr()* fonksiyonları, tıpkı *strstr()* gibi arama için kullanılır. Ancak *strstr()* fonksiyonu katar içinde bir başka katarı arayabilirken, *strchr()* ve *strrchr()* fonksiyonları katar içinde tek bir karakter aramak için kullanılır. *strchr()*, karakterin katar içindeki ilk konumunu gösterirken; *strrchr()* fonksiyonu, ilgili karakterin son kez geçtiği adresi verir.

```
#include<stdio.h>
#include<string.h>
int main( void )
{
    char adres[] = "Esentepe Caddesi Mecidiyekoy Istanbul";
    char *ilk_nokta, *son_nokta;
    ilk_nokta = strchr( adres, 'e' );
    son_nokta = strrchr( adres, 'e' );
    if( ilk_nokta != NULL ) {
        printf( "Ilk gorundugu konum: %d\n", ilk_nokta - adres
    );
        printf( "Son gorundugu konum: %d\n", son_nokta - adres
    );
    }
    else
        printf( "Eşleşme bulunamadı.\n" );
    return 0;
}
```

* atoi() ve atof() ile katar dönüşümü

Verilen katarı, sayıya çevirmek gerekebilir. Eğer elinizdeki metni, bir tam sayıya (*int*) çevirecekseniz, *atoi()* fonksiyonunu kullanmanız gerekir. Şayet dönüşüm sonunda elde etmek istediğiniz değişken tipi, virgüllü sayı ise (*float*), *atof()* fonksiyonu kullanılır. Her iki fonksiyon *stdlib.h* kütüphanesi içindedir. Bu fonksiyonları kullanırken, *#include<stdlib.h>* komutunu program başlangıcına yazmalısınız.

```
#include<stdio.h>
#include<stdlib.h>
int main( void )
{
    char kok_iki[] = "1.414213";
    char pi[] = "3.14";
    char tam_bir_sayi[] = "156";
    char hayatın_anlami[] = "42 is the answer";

    printf( "%d\n", atoi( tam_bir_sayi ) );
    printf( "%d\n", atoi( hayatın_anlami ) );
    printf( "%f\n", atof( kok_iki ) );
    printf( "%f\n", atof( pi ) );
    return 0;
}
```

Her iki fonksiyonda rakam harici bir şey görene kadar çalışır. Eğer nümerik ifadeler dışında bir karakter çıkarsa, fonksiyon o noktada çalışmayı keser.

Örnek Sorular

Soru 1: Kendisine verilen bir katarın boyutunu bulan fonksiyonu yazınız. (Çözüm için *strlen()* fonksiyonunu kullanmayınız.)

```
#include<stdio.h>
#include<string.h>
int katar_boyutu_bul( char [] );
int main( void )
{
    char test_katari[50];
    strcpy( test_katari, "ABCDEF" );
    printf( "Katar boyutu: %d\n", katar_boyutu_bul( test_katari ) );
    return 0;
}
int katar_boyutu_bul( char katar[] )
{
    int i;
    for( i = 0; katar[ i ]!='\0'; i++ );

    return i;
}
```

Soru 2: Tersinden de aynı şekilde okunabilen kelime, cümle veya mısraya '*palindrome*' denmektedir. Adı *palindrome()* olan ve verilen katarın tersinin kendisine eşit olduğu durumda geriye 1; aksi hâlde 0 döndüren fonksiyonu yazınız.

```
#include<stdio.h>
#include<string.h>
int palindrome( char [] );
int main( void )
{
    char test_katari[50];
    strcpy( test_katari, "ABBA" );
    printf( "%d\n", palindrome( test_katari ) );
    return 0;
}
int palindrome( char katar[] ){
    int boyut =0 , i;
    // Once katar boyutu bulunuyor
    for( boyut = 0; katar[ boyut ]!='\0'; boyut++ );
    for( i = 0; i < boyut/2; i++ ) {
        if( katar[i] != katar[ boyut - i - 1 ] )
            return 0;
    }
    return 1;}
}
```

Soru 3: Aşağıdaki gibi çalışıp, çıktı üretebilecek *"ters_cevir"* programını oluşturunuz.

```
$ ./ters_cevir Merhaba Dunya Nasilsin?  
abahreM aynuD ?nislisaN
```

```
#include<stdio.h>  
#include<string.h>  
void ters_cevir( char [] );  
int main( int argc, int arg[] )  
{  
    int i;  
    for( i = 1; i < argc; i++ ) {  
        ters_cevir( arg[i] );  
    }  
    printf("\n");  
    return 0;  
}  
void ters_cevir( char katar[] )  
{  
    int i, boyut;  
    for( boyut = 0; katar[ boyut ] != '\0'; boyut++ );  
  
    for( i = 0; i < boyut; i++ )  
        printf("%c", katar[ boyut - 1 - i ] );  
    printf(" ");  
}
```

İç içe for döngüsü örneği (eşkenar dörtgen oluşturma)

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

int main()
{
    int i,j;
    for(i=1;i<10;i+=2) //kaç satır olacak? 9 satır olarak verilmiş
    {
        for(j=0;j<9-i/2;j++) // bosluk sayısı
        {
            printf(" "); }
        for(j = 0; j < i; j++)
        {
            printf("*"); } // yıldız sayısı
        printf("\n");
    }

    for(i=7;i>0; i -=2) {

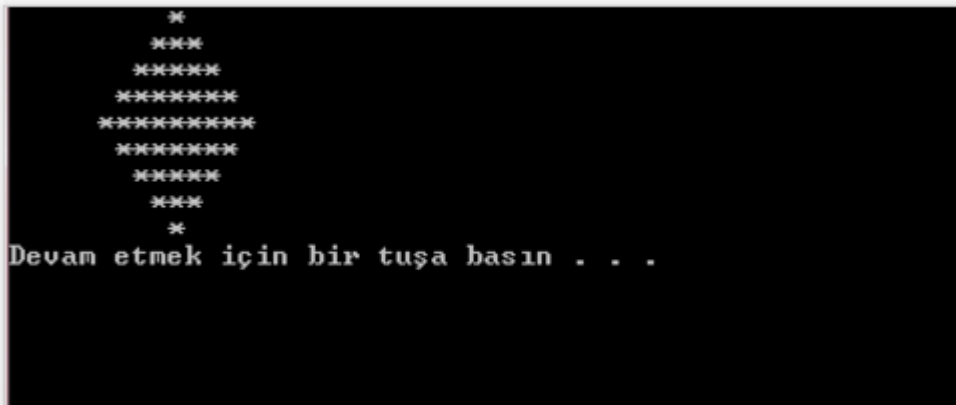
        for(j = 0; j < 9 - i / 2; j++)
        {
            printf(" ");
        }

        for(j = 0; j < i; j++)
        {
            printf("*");
        }

        printf("\n");
    }

    system("PAUSE");
    return 0;
}
```

9th July 2012, Aslıhan-Kaymaz tarafından yayınlandı



Rastgele Sayı Üretme

```

#include<conio.h>
#include<stdio.h>
main(){
int i;
for (i=0;i<5;i++)
printf("%d ",rand());
getch();return 0;}
//her çalıştığında rastgele aynı sayıları üretiyor
#include<stdio.h>
#include<stdlib.h>
main(){
int a=0,i=0;
for (i=0;i<5;i++)
{
a=rand();
printf("\n%d",a);
}
system("pause");return 0;}
her çalıştığında farklı sayılar üretir ancak
srand(2), srand(3),
srand(4) yapmak lazım,
#include<stdio.h>
#include<stdlib.h>
main(){
int a=0,i=0;
srand(2);
for (i=0;i<5;i++)
{
a=rand();
printf("\n%d",a);
}
system("pause");return 0;}

```

srand(x) xi sürekli değiştirmek gerekiyor bunun yerine x=time(NULL) kullanırız bu zamanda bağlı değer değiştiren komuttur

```
#include<stdio.h>
#include<stdlib.h> /*rand kütüphaneye ait*/
#include<conio.h>
#include<time.h> /*time(NULL) kütüphaneye ait*/
main(){
int a=0,i=0,sayi;
sayi=time(NULL);
srand(sayi);
for (i=0;i<5;i++)
{
a=rand();
printf("\n%d",a);
}
getch();return 0;}
```

%%%% üretilen sayının istenilen aralıkta olmasını istersek mod kullanırız örneğin 1-6 arası rastgele değer üretelim mod 6 ile 0-5 arası rastgele sayı üretilir+1 eklersek 1-6 arası sayı üretmiş oluruz

```
#include<stdio.h>
#include<stdlib.h> /*rand kütüphaneye ait*/
#include<conio.h>
#include<time.h> /*time(NULL) kütüphaneye ait*/
main(){
int sayi=0;
srand(time(NULL));
sayi=(rand()%6)+1;
printf("\nuretilen sayi %d",sayi);
getch();return 0;}
```

Örneğin rastgele bir sayı üretelim. Sayımız 0-10 arası bir sayı olsun. Rand() komutu yalnız kullanıldığında;

```
#include<-stdio.h>
#include<-conio.h>
#include<-stdlib.h>
int main(){
int sayi;
sayi=rand()%10;
printf("Rastgele sayi= %d",sayi);
getch();}
```

Örnek Aynı işlemi birde srand komutu ile birlikte kullanalım. Bu komutun asıl görevi zamana bağlı bir değer olarak sürekli olarak üretilen sayının değişmesini sağlar. Yani zaman sürekli değişen bir değerdir. Sayıda sürekli değişsin diye bu komutu tanımlama bölümünden sonra kullanırsanız, az önce sabit değer alan sayı değişkenimiz sürekli değişecektir.

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
int main(){
    int sayi;
    srand(time(NULL));
    sayi=rand()%11;
    printf("Rastgele sayi= %d",sayi);
    getch();return 0;}
```

Bu kısım da yine rastgele bir sayı üretir. Ama az önce söylediğimiz gibi bu üretilen sayıyı zaman gibi sürekli değişen bir değere bağlayarak sürekli değişmesini sağlayabiliriz. Srand(time(NULL)) komutu gördüğünüz gibi tanımlama kısmından sonra kullanılarak sayı değişkeni için üretilen sayının sürekli olarak değişmesini sağlar. Bir kere tanımlamak yapmak yeterlidir. Kullanımı aynen örnekteki gibidir. NULL nedir diye merak eden varsa, programın istenilen sürekli olarak değişmesini ve çalışmasını sağlar. Not: Üretilen sayı en az 0'dır. Bunun nedenide kalanın hiç bir zaman negatif olamamasıdır.

Diğer örnekler:

A) 0-25 arası rastgele 10 adet sayı üretmek:

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main(){
    int a=1,sayi;
    srand(time(NULL));
    while(a<=10){
        sayi=rand()%25;
        printf("%d sayi=%dn",a,sayi);
        a++; }        getch(); }
```

B) Milli Piyango bileti yapma:

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main(){
    int a,b;
    srand(time(NULL));
    printf("***Milli Piyango***n");
    for(a=1;a<=6;a++){
        b=rand()%10;
        printf("%dt",b); }
    getch(); }
```

C) Otomatik sayısal loto kuponu dolduran program

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main()
{
    int a,b=1,sayi;
    srand(time(NULL));
    printf("***sayısal loto***n");
    while(b<=6){
        printf("%d.kolont",b);
        for(a=1;a<=6;a++){
            sayi=rand()%50;
            printf("t%d ",sayi);}
        b++;
        printf("tn");
        getch();
    }
}
```

D) İstenilen adette atılan zarın her atış sonrası değerini bulan program:

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main(){
    int a,zar,atis;
    srand(time(NULL));
    printf("---Zar atma---n");
    printf("Zarın kaç kere atılacağını girin= ");
    scanf("%d",&zar);

    for(a=1;a<=zar;a++){
        don:
        atis=rand()%7;
        if(atis!=0){
            printf("%d.atis=%dn",a,atis);
        }
        else{
            goto don;
        }
    }
    getch();
}
```

E) Çarpmayı yeni öğrenenler için çalışma programı:

Çalışma Mantığı: Doğru cevapta yeni soru hatada ise bilemediniz diyerek program biter:

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main(){
    int a,b,c,d;
    srand(time(NULL));
    for(a=1;a<=100;a++){
        b=rand()%10;
        c=rand()%10;
        printf("%d*%d= ",b,c);
        scanf("%d",&d);
        if (d==b*c){
            printf("aferin, dogru...n");
        }
        else{
            printf("bilemedin...");
            break;
        } }
    getch(); }
```

F) Barbut oyunu: Oyuncu sayısı girilerek her kişi için atılan zar değerini yazan

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main()
{
    int a,b,c;
    srand(time(NULL));
    printf("----Barbut Oyunu----n");
    printf("Oyuncu sayisi gir=");
    scanf("%d",&a);
    for(b=1;b<=a;b++){
        don:
        c=rand()%7;
        if(c!=0){
            printf("%d.kisi=%dt",b,c);
        }
        else{
            goto don;}
        }
    getch();
    }
```

PROF. DR. ALİ ÖZTÜRK

G) Bir markette seçilen kart sayısı ile türetilen sayı eşit ise kullanıcıya "Tebrikler kazandınız..." diyen diğer durumda tekrar seçim yaptıran...

```
#include<-stdio.h>
#include<-stdlib.h>
#include<-conio.h>
#include<-time.h>
int main()
{
    int kart,b;
    srand(time(NULL));
    tekrar:
    printf("0-10 arasi bir kart nosu giriniz: ");
    scanf("%d",&kart);
    b=rand()%11;
    if(kart==b){
        printf("Tebrikler Kazandınız...");
    }
    else{
        printf("Tekrar Deneyin....n");
        goto tekrar;
    }
    getch(); }
```

DOSYA İŞLEMLERİ

Ornek1 devcpp klasörüne deneme.txt formatında dosya açılacak ve içine hello World yazdırılacak

```
#include <stdio.h>

//#include <conio.h>

main(){

    FILE *dosya;

    dosya=fopen("deneme.txt","w") ;

    fputs("hello world",dosya);

    fclose(dosya);

    //getch(); return 0;}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Ornek2 Devcpp klasorunun içine ali.txt uzantılı bir dosya açtık ve dosya içine bu dosya Ali nin dosyasıdır yazısını yazdırdık

```
#include <stdio.h>

//#include <conio.h>

main(){

    FILE *dosya;

    char isim[25]="ali";

    dosya = fopen("ali.txt","w") ;

    fprintf(dosya, "bu dosya %s nin dosyasıdır",isim);

    fclose(dosya);

    //getch(); return 0;}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Ornek3

```
#include <stdio.h>

main(){

    FILE *dosya;

    dosya = fopen("carpim tablosu.txt","w") ;

    int i,j ;

    for(i=1;i<=10;i++){
        for(j=1;j<=10;j++){
            fprintf(dosya, "%d x%d = %d\n",i,j,i*j);
        }
        fprintf(dosya, "\n");
    }
    fclose(dosya);
}
```

%%%%%%%%%

Ornek 4

```
#include <stdio.h>

//#include <conio.h>

main(){

    FILE *dosya;

    dosya = fopen("ogrenci.txt","w") ;

    char isim[20], okul[20], bolum[50];

    printf("adınız:"); gets(isim);

    printf("okulunuz:"); gets(okul);

    printf("bolumunuz:"); gets(bolum);

    fprintf(dosya,"%s \t %s \t %s",isim,okul,bolum);

    fprintf(dosya, "\n");

    fclose(dosya);

    //getch(); return 0;}

%%%%%%%%%
```

Ornek 5

```
#include <stdio.h>

//#include <conio.h>

main(){

    FILE *dosya;

    dosya=fopen("merhaba.txt","a");

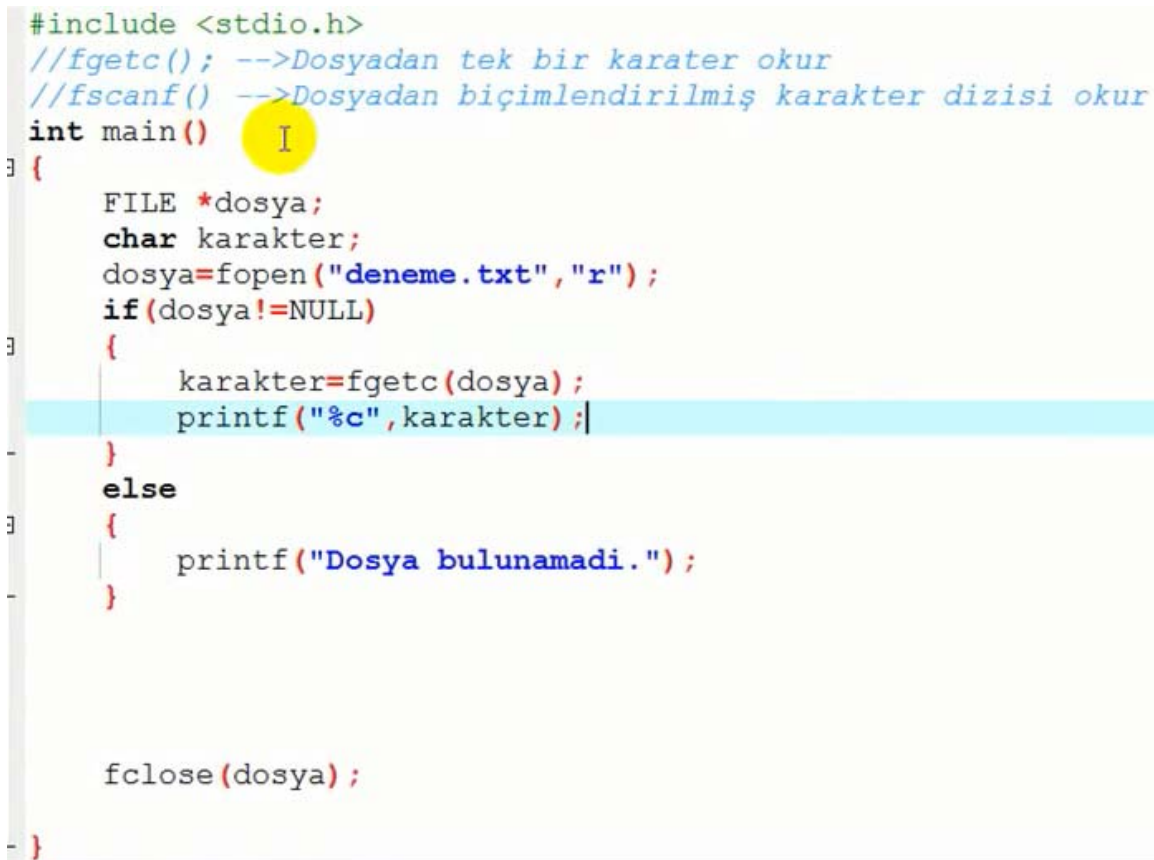
    fputs("\n merhaba dunya",dosya);

    fclose(dosya);

    //getch(); return 0;

}
```

Örnek , Devcpp klasörü içine deneme.txt uzantılı dosya aç ve içine Ali yaz oradan A harfini alacaz



```
#include <stdio.h>
//fgetc(); -->Dosyadan tek bir karater okur
//fscanf() -->Dosyadan biçimlendirilmiş karakter dizisi okur
int main()
{
    FILE *dosya;
    char karakter;
    dosya=fopen("deneme.txt", "r");
    if (dosya!=NULL)
    {
        karakter=fgetc(dosya);
        printf("%c", karakter);
    }
    else
    {
        printf("Dosya bulunamadi.");
    }

    fclose(dosya);
}
```

Deneme txt dosyasına Merhaba Ali 23 yaz onu alacaz

Şimdi programa gecelim ;

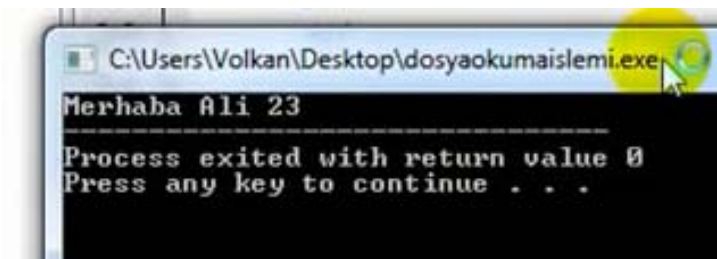
```
#include <stdio.h>
//fgetc(); -->Dosyadan tek bir karakter okur
//fscanf() -->Dosyadan biçimlendirilmiş karakter dizisi okur
int main()
{
    FILE *dosya;
```

```
    char k1[10],k2[10];
    int sayi;

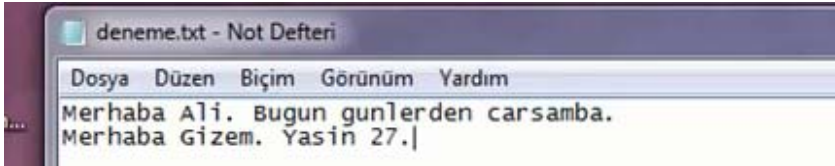
    if((dosya=fopen("deneme.txt","r"))!=NULL)
    {
        fscanf(dosya,"%s",&k1);
        fscanf(dosya,"%s",&k2);
        fscanf(dosya,"%d",&sayi);
        printf("%s %s %d",k1,k2,sayi);
    }
    else{
        printf("Dosya bulunamadi..");
    }

    fclose(dosya);
}
```

Ekran çıktısı;



Örnek, bir cümle yazalım 50 kelime her kelime max 20 karakter olsun bir dizi tanımlarız feof dosyanın sonuna gelinip gelinmediğini kontrol eder



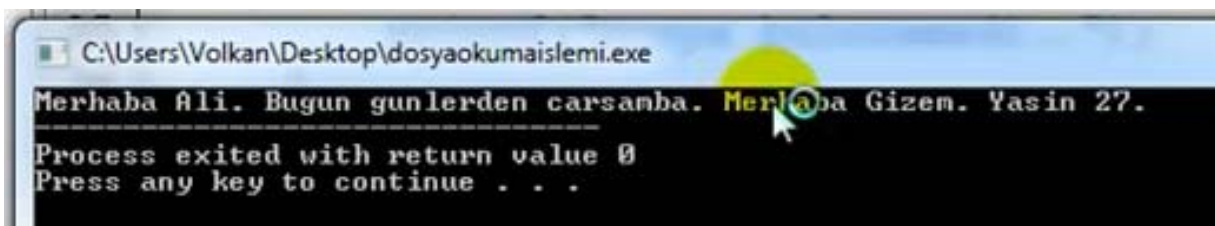
Program kodu,

```
#include <stdio.h>
//fgetc(); -->Dosyadan tek bir karakter okur
//fscanf() -->Dosyadan biçimlendirilmiş karakter dizisi okur
int main()
{
    FILE *dosya;

    char kelime[50][20];
    int i=0;
    if((dosya=fopen("deneme.txt", "r")) != NULL)
    {
        while(!feof(dosya))
        {
            fscanf(dosya, "%s", &kelime[i]);
            printf("%s ", kelime[i]);
            i++;
        }
    }
    else
    {
        printf("Dosya bulunamadi..");
    }

    fclose(dosya);
}
```

Ekran çıktısı



PROF. DR. ALİ ÖZTÜRK

Örnek Öğrenci numara isim notlar dosyada olsun devc içinde deneme. Txt uzantılı dosya aç listeyi içine yaz 10 öğrenci numara max 10 hane 10 öğrenci ismi isim max 20 harf olsun

While (!feof(dosya)) dosya sonuna gelinmediği sürece bu işlemi yap demek

Dosya	Düzen	Biçim	Görünüm	Yar
2248	Ahmet	45		
1823	Meltem	88		
9485	Birsen	79		
4681	Mehmet	92		
3123	Tuncay	0		
9875	Kemal	73		
5576	Berk	0		
2563	Hakan	54		
6548	Cemil	38		
4591	Hasan	77		

Program kodu

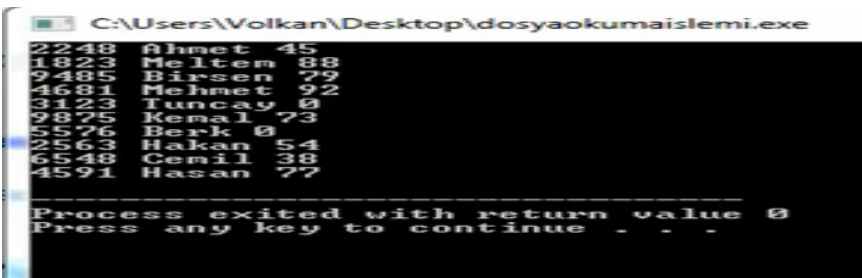
```
#include <stdio.h>
//fgetc(); -->Dosyadan tek bir karakter okur
//fscanf() -->Dosyadan biçimlendirilmiş karakter dizisi okur
int main()
{
    FILE *dosya;

    char numaralar[10][10], isimler[10][20];
    int notlar[10], i=0;

    if((dosya=fopen("deneme.txt", "r")) != NULL)
    {
        while(!feof(dosya))
        {
            fscanf(dosya, "%s %s %d", &numaralar[i], &isimler[i], &notlar[i]);
            printf("%s %s %d\n", numaralar[i], isimler[i], notlar[i]); i++;
        }
    }
    else
    {
        printf("Dosya Bulunamadi.");
    }

    fclose(dosya);
}
```

ekran çıktısı



```
2248 Ahmet 45
1823 Meltem 88
9485 Birsen 79
4681 Mehmet 92
3123 Tuncay 0
9875 Kemal 73
5576 Berk 0
2563 Hakan 54
6548 Cemil 38
4591 Hasan 77

Process exited with return value 0
Press any key to continue . . .
```

Örnek aynı deneme.txt dosyasına bir şiir yazalım dosyadaki son karaktere kadar okuma yapsın While (karakter!= EOF) karakter son karakter olmadığı sürece işlemi yap demek

```
Bir şiir
Tek bir şiir yazmalıyım
Uyağı rüzgâr olan
Yağmura bürünmüş soluğu

Bir gün
Tek bir gün kalmalı
Benden kalacaksa geriye
Bir öpüş tadı dudağımda

ve bir öpüş tadında
Olmalı o şiir de
```

Program kodu

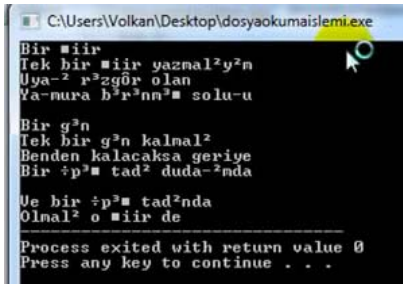
```
#include <stdio.h>
//fgetc(); -->Dosyadan tek bir karakter okur
//fscanf() -->Dosyadan biçimlendirilmiş karakter dizisi okur
int main()
{
    FILE *dosya;

    char karakter;

    if ((dosya=fopen("deneme.txt", "r")) != NULL)
    {
        karakter=fgetc(dosya);
        while (karakter!=EOF)
        {
            printf("%c", karakter);
            karakter=fgetc(dosya);
        }
    }
    else
    {
        printf("Dosya bulunamadi");
    }

    fclose(dosya);
}
```

Ekran çıktısı ,



```
C:\Users\Volkan\Desktop\dosyaokumaislemi.exe
Bir miiir
Tek bir miiir yazmal²y²n
Uya-² r²zgôr olan
Ya-nura b²r²nn² solu-u

Bir g²n
Tek bir g²n kalmal²
Benden kalacaksa geriye
Bir ÷p² tad² duda-²nda

ve bir ÷p² tad²nda
Olmal² o miiir de

Process exited with return value 0
Press any key to continue . . .
```


Ornek

```
#include <stdio.h>

main() {
    FILE *d1, *d2;

    d1 = fopen("personelSeq.txt", "r");
    d2 = fopen("personelRnd.txt", "w");

    while(1) {

        fscanf(d1, "%d %s %s %s %f", &kayit.perno,
            kayit.ad, kayit.soyad, kayit.tel, &kayit.maas);

        if(feof(d1)) break;

        fwrite(&kayit, sizeof(kayit), 1, d2);

    }

    fclose(d1); fclose(d2);

}
```

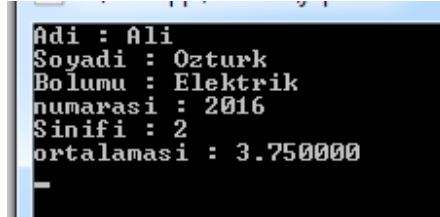
YAPI BİRLİK

ÖRNEK 1

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct ogrenci
{
    char ad[20];
    char soyad[20];
    char bolum[50];
    int numara;
    int sinif;
    float ortalama;
    }ogr, ogr1,ogr2;
```

```
main(){
strcpy(ogr.ad,"Ali");
strcpy(ogr.soyad,"Ozturk");
strcpy(ogr.bolum,"Elektrik");
ogr.numara=2016;
ogr.sinif=2;
ogr.ortalama=3.75;
printf("Adi : %s\n",ogr.ad);
printf("Soyadi : %s\n",ogr.soyad);
printf("Bolumu : %s\n",ogr.bolum);
printf("numarasi : %d\n",ogr.numara);
printf("Sinifi : %d\n",ogr.sinif);
printf("ortalamasi : %f\n",ogr.ortalama);
```

```
getch();return 0;}
```



```
Adi : Ali
Soyadi : Ozturk
Bolumu : Elektrik
numarasi : 2016
Sinifi : 2
ortalamasi : 3.750000
```

Örnek2

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

```
struct ogrenci
{
    char ad[20];
    char soyad[20];
    char bolum[50];
    int numara;
    int sinif;
    float ortalama;
}ogr, ogr1,ogr2;
```

```
main(){
    strcpy(ogr.ad,"Ali");
    strcpy(ogr.soyad,"Ozturk");
    strcpy(ogr.bolum,"Elektrik");
    ogr.numara=2016;
    ogr.sinif=2;
    ogr.ortalama=3.75;
    printf("Adi : %s\n",ogr.ad);
    printf("Soyadi : %s\n",ogr.soyad);
    printf("Bolumu : %s\n",ogr.bolum);
    printf("numarasi : %d\n",ogr.numara);
    printf("Sinifi : %d\n",ogr.sinif);
    printf("ortalamasi : %f\n",ogr.ortalama);
```

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

```
strcpy(ogr.ad,"Veli");
strcpy(ogr.soyad,"Yildiz");
strcpy(ogr.bolum,"Elektronik");
ogr.numara=2214,
ogr.sinif=4;
ogr.ortalama=1.75;
printf("Adi : %s\n",ogr.ad);
printf("Soyadi : %s\n",ogr.soyad);
printf("Bolumu : %s\n",ogr.bolum);
printf("numarasi : %d\n",ogr.numara);
printf("Sinifi : %d\n",ogr.sinif);
printf("ortalamasi : %f\n",ogr.ortalama);
```

```
getch();return 0;}
```

PROF. DR. ALİ ÖZTÜRK

```
Adi : Ali
Soyadi : Ozturk
Bolumu : Elektrik
numarasi : 2016
Sinifi : 2
ortalamasi : 3.750000
```

```
Adi : Veli
Soyadi : Yildiz
Bolumu : Elektronik
numarasi : 2214
Sinifi : 4
ortalamasi : 1.750000
```

Örnek3

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

```
struct kitaplar
{
    char ad[50];
    char yazar[50];
    float fiyat;
}
kitap1={"c program","Ali Ozturk",28.75},
kitap2={"matematik","Melih Ates",48.75},
kitap3={"Arabesk","Muslum Gurses",4.25};
```

```
main(){
```

```
printf("Kitap Adı: %s\n",kitap1.ad);
printf("Yazar Adı: %s\n",kitap1.yazar);
printf("Fiyatı: %f\n",kitap1.fiyat);
```

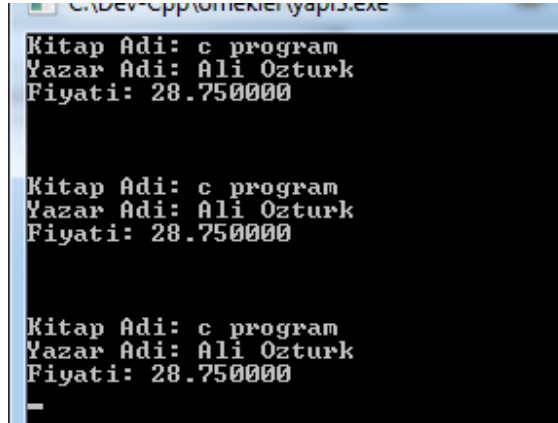
```
printf("\n\n\n");
```

```
printf("Kitap Adı: %s\n",kitap1.ad);
printf("Yazar Adı: %s\n",kitap1.yazar);
printf("Fiyatı: %f\n",kitap1.fiyat);
```

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

```
printf("Kitap Adı: %s\n",kitap1.ad);
printf("Yazar Adı: %s\n",kitap1.yazar);
printf("Fiyatı: %f\n",kitap1.fiyat);
```

```
getch();return 0;}
```



```
Kitap Adı: c program
Yazar Adı: Ali Ozturk
Fiyati: 28.750000

Kitap Adı: c program
Yazar Adı: Ali Ozturk
Fiyati: 28.750000

Kitap Adı: c program
Yazar Adı: Ali Ozturk
Fiyati: 28.750000
```

Örnek 4 struct dizi kullanımı

```

#include<stdio.h>
#include<conio.h>
// struct in dizi ile kullanımı
struct ogrenciler
{
    char ad[50];
    char bolum[50];
    int sinif;
    float ortalama;
}

ogr[3]={
    {"Ali","Elektrik",2,3.75},
    {"Veli","Makina",3,2.25},
    {"Can","Bilgisayar",4,2.77}
};

main(){

printf("Adi: %s\n",ogr[0].ad);
printf("Bolumu: %s\n",ogr[0].bolum);
printf("sinifi: %d\n",ogr[0].sinif);
printf("Ortalamasi: %f\n",ogr[0].ortalama);

printf("\n\n\n");

printf("Adi: %s\n",ogr[1].ad);
printf("Bolumu: %s\n",ogr[1].bolum);
printf("sinifi: %d\n",ogr[1].sinif);
printf("Ortalamasi: %f\n",ogr[1].ortalama);

printf("\n\n\n");

printf("Adi: %s\n",ogr[2].ad);
printf("Bolumu: %s\n",ogr[2].bolum);
printf("sinifi: %d\n",ogr[2].sinif);
printf("Ortalamasi: %f\n",ogr[2].ortalama);

getch();return 0;}

```

```

Adi: Ali
Bolumu: Elektrik
sinifi: 2
Ortalamasi: 3.750000

Adi: Veli
Bolumu: Makina
sinifi: 3
Ortalamasi: 2.250000

Adi: Can
Bolumu: Bilgisayar
sinifi: 4
Ortalamasi: 2.770000

```

Örnek 5 Atama işlemleri main içinde de yapılabilir

```
#include<stdio.h>
#include<conio.h>
// struct in dizi ile kullanımı
// atama main icinde de olabilir
struct ogrenciler
{
    char ad[50];
    char bolum[50];
    int sinifi;
    float ortalama;
};

main(){

    struct ogrenciler ogr[3]={
        {"Ali","Elektrik",2,3.75},
        {"Veli","Makina",3,2.25},
        {"Can","Bilgisayar",4,2.77}
    };

    printf("Adi: %s\n",ogr[0].ad);
    printf("Bolumu: %s\n",ogr[0].bolum);
    printf("sinifi: %d\n",ogr[0].sinif);
    printf("Ortalamasi: %f\n",ogr[0].ortalama);

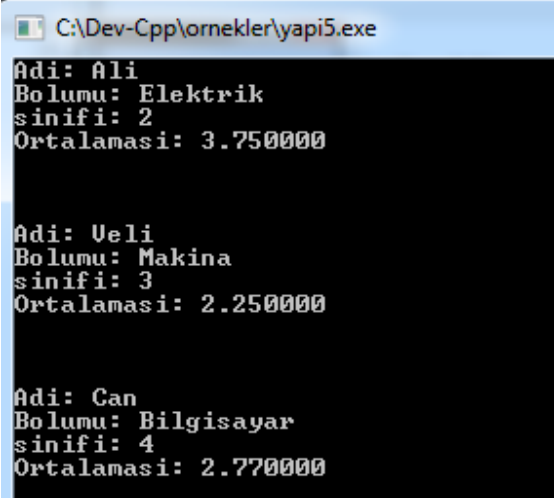
    printf("\n\n\n");

    printf("Adi: %s\n",ogr[1].ad);
    printf("Bolumu: %s\n",ogr[1].bolum);
    printf("sinifi: %d\n",ogr[1].sinif);
    printf("Ortalamasi: %f\n",ogr[1].ortalama);

    printf("\n\n\n");

    printf("Adi: %s\n",ogr[2].ad);
    printf("Bolumu: %s\n",ogr[2].bolum);
    printf("sinifi: %d\n",ogr[2].sinif);
    printf("Ortalamasi: %f\n",ogr[2].ortalama);

    getch();return 0;}
```



```
C:\Dev-Cpp\ornekler\yapi5.exe
Adi: Ali
Bolumu: Elektrik
sinifi: 2
Ortalamasi: 3.750000

Adi: Veli
Bolumu: Makina
sinifi: 3
Ortalamasi: 2.250000

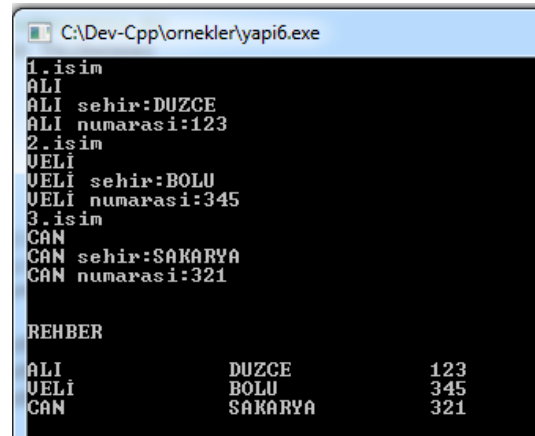
Adi: Can
Bolumu: Bilgisayar
sinifi: 4
Ortalamasi: 2.770000
```

Örnek 6 kullanıcıdan alınan değerler struct gönder ve ekrana bas

```
#include<stdio.h>
#include<conio.h>
struct TelRehber
{
    char ad[50];
    char sehir[50];
    int tel;
} tel[5];

main(){
    int i;
    for(i=1;i<=3;i++)
    {
        printf("%d.isim\n",i); scanf("%s",&tel[i].ad);
        printf("%s sehir:",tel[i].ad); scanf("%s",&tel[i].sehir);
        printf("%s numarasi:",tel[i].ad); scanf("%d",&tel[i].tel);
    }
    printf("\n\nREHBER\n\n");
    for(i=1;i<=3;i++){

        printf("%s\t\t%s\t\t%d\n",tel[i].ad,tel[i].sehir,tel[i].tel) ;
    }
    getch();return 0;}
```



```
C:\Dev-Cpp\ornekler\yapi6.exe
1.isim
ALI
ALI sehir:DUZCE
ALI numarasi:123
2.isim
UELİ
UELİ sehir:BOLU
UELİ numarasi:345
3.isim
CAN
CAN sehir:SAKARYA
CAN numarasi:321

REHBER

ALI          DUZCE          123
UELİ         BOLU           345
CAN          SAKARYA        321
```

ÖRNEK7 Struct pointer Gösterimi

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
struct futbolcu
{
    char ad[50];
    char takim[50];
    int yas;
};
main(){

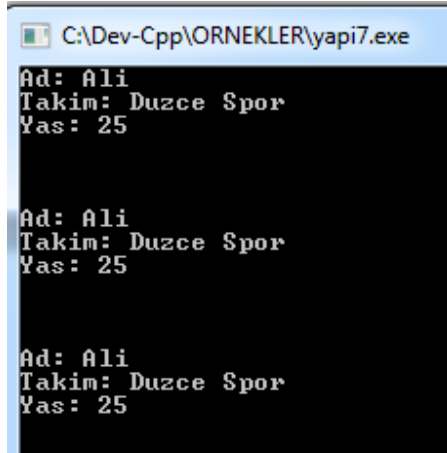
struct futbolcu f;
strcpy(f.ad,"Ali");
strcpy(f.takim,"Duzce Spor");
f.yas=25;

printf("Ad: %s\n",f.ad);
printf("Takim: %s\n",f.takim);
printf("Yas: %d\n",f.yas);

printf("\n\n\n");
struct futbolcu *fptr=&f;
printf("Ad: %s\n",(*fptr).ad);
printf("Takim: %s\n",(*fptr).takim);
printf("Yas: %d\n",(*fptr).yas);
printf("\n\n\n");
printf("Ad: %s\n",fptr->ad);
printf("Takim: %s\n",fptr->takim);
printf("Yas: %d\n",fptr->yas);

getch();return 0;}

```



```

C:\Dev-Cpp\ORNEKLER\yapi7.exe
Ad: Ali
Takim: Duzce Spor
Yas: 25

Ad: Ali
Takim: Duzce Spor
Yas: 25

Ad: Ali
Takim: Duzce Spor
Yas: 25

```


ÖRNEK 8

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct sehirler
{
    char ad[50];
    int nufus;
};

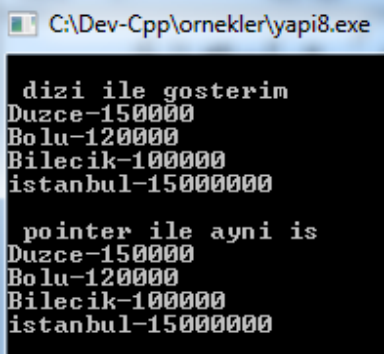
main(){
    printf("\n dizi ile gosterim\n");
    struct sehirler s[4]={
        {"Duzce",150000},
        {"Bolu",120000},
        {"Bilecik",100000},
        {"istanbul",15000000},
    };
    int i;
    for(i=0;i<4;i++){
        printf("%s-%d\n",s[i].ad,s[i].nufus);

    }
```

```
    struct sehirler *sptr=s;
    printf("\n pointer ile ayni is \n");

    for(i=0;i<4;i++){

        printf("%s-%d\n",sptr->ad,sptr->nufus);
        sptr++;
    }
    getch();return 0;}
```



```
C:\Dev-Cpp\ornekler\yapi8.exe

dizi ile gosterim
Duzce-150000
Bolu-120000
Bilecik-100000
istanbul-15000000

pointer ile ayni is
Duzce-150000
Bolu-120000
Bilecik-100000
istanbul-15000000
```