

CENG 111 ALGORİTMALAR VE PROGRAMLAMA

Prof. Dr. Tufan TURACI

tturaci@pau.edu.tr

- Pamukkale Üniversitesi
- Mühendislik Fakültesi
- Bilgisayar Mühendisliği Bölümü
- Hafta 13

13. Hafta Konular

--- Yapılar (Structures)

--- Dosyalar (Files)

Yapılar (Structures)

- Birbirleriyle bağlantılı ve bir küme teşkil eden değerler bir tek değişken çatısı altında bu değişkenin alt alanları olarak tanımlanabilir.
- Bu durumda bir grup değere bir tek isimle ulaşılabilir.
- Bu yeni tanımlanan değişken yapı (struct) olarak ifade edilir.
- struct tipi değişkenler birden fazla tek tipte olabilen verilerin oluşturduğu bir bütündür.
- Genel Kullanımı aşağıdaki şekildedir:

```
struct yapı_adı  
{ tip değişken_ismi 1;  
  tip değişken_ismi 2;  
  .  
  .  
  tip değişken_ismi n;  
};
```

- Yapılar dizilere benzetilebilir. Çünkü ikisinde de birbiriyle ilgili olan veriler içerilir.
- Farkları, dizilerde verilen aynı tiptedir. Fakat yapılar birbirinden farklı tipte veriler içerebilir. Bu sayede veriler daha düzenli bir şekilde temsil edilirler.

Örnek: Aşağıda bir öğrencinin numarası, adı, soyadı, vize notu ve final notunu içeren bir yapı tanımlanmıştır.

```
struct Ogrenci
{
int No;
char Ad[50];
char Soyad[50];
int Vize;
int Final;
};
```



Yapı tanımlandı, yandaki şekilde kullanılır.

```
int main()
{
struct Ogrenci ogrenci_bilgi;
}
```

int main() deki tanımlamadan sonra verilere aşağıdaki şekilde ulaşılır:

ogrenci_bilgi.No

ogrenci_bilgi.Ad

ogrenci_bilgi.Soyad

ogrenci_bilgi.Vize

ogrenci_bilgi.Final

C kodu:

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

```
struct Ogrenci
```

```
{
```

```
int No;
```

```
char Ad[50];
```

```
char Soyad[50];
```

```
int Vize;
```

```
int Final;
```

```
};
```

```
int main()
```

```
{
```

```
struct Ogrenci ogrenci_bilgi;
```

```
printf("*****\n VERI GIRISI\n*****\n");
```

```
printf("Ogrenci No:");
```

```
scanf("%d",&ogrenci_bilgi.No);
```

```
printf("Ogrenci Ad:");
```

```
scanf("%s",ogrenci_bilgi.Ad);
```

```
printf("Ogrenci Soyad:");
```

```
scanf("%s",ogrenci_bilgi.Soyad);
```

```
printf("Ogrenci Vize Notu:");
```

```
scanf("%d",&ogrenci_bilgi.Vize);
```

```
printf("Ogrenci Final Notu:");
```

```
scanf("%d",&ogrenci_bilgi.Final);
```





```
printf("*****\n VERI YAZMA\n*****\n");
//printf("No: %d", ogrenci_bilgi.No);
//printf("Ad: %s", ogrenci_bilgi.Ad);
printf("No: %d \nAd: %s\nSoyad: %s\n Vize: %d\n Final: %d\n",
ogrenci_bilgi.No,
ogrenci_bilgi.Ad,
ogrenci_bilgi.Soyad,
ogrenci_bilgi.Vize,
ogrenci_bilgi.Final
);
getch ();
return 0;
}
```

```
*****
VERI GIRISI
*****
Ogrenci No:20210013
Ogrenci Ad:Ali
Ogrenci Soyad:Can
Ogrenci Vize Notu:65
Ogrenci Final Notu:90
*****
VERI YAZMA
*****
No: 20210013
Ad: Ali
Soyad: Can
Vize: 65
Final: 90
-----
```

- Yapılar diziler gibi bellekte sürekli kalır.
- Bir yapı içerisindeki elemanlara üye(member) denir.
- Üyelerin her biri farklı veri tipine sahip olabilir.
- Yapılar sayesinde kendi veri tipinizi üretmeniz mümkündür.
- Yapılar farklı programlama dillerinde Record(Kayıt) olarak da isimlendirilirler.
- Yapılar, nesne tabanlı programlamanın da temelini oluşturan bir yaklaşımdır.
- Yapılar, C# ve Java gibi tamamen nesneye dayalı programlamayı benimsemiş gelişmiş dilleri öğrenmeye de yardımcı olacaktır.

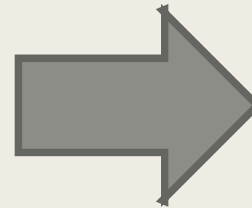
struct Değişkenlerde Atama

- struct tipindeki bir değişkenin değeri aynı tipteki bir başka struct değişkenine atanabilir.
- Üyeleri tek tek atmaya gerek yoktur.
- Bir struct değişken başka bir struct değişkene aşağıdaki şekilde atanır:

öğrenci.bilgi1 = öğrenci.bilgi2;

```
int main()
{
    struct Ogrenci ogrenci_bilgi1, ogrenci_bilgi2;

}
```



ogrenci_bilgi1 ve ogrenci_bilgi2
soldaki şekilde tanımlanmıştır.

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

```
struct Ogrenci
```

```
{
```

```
int No;
```

```
char Ad[50];
```

```
char Soyad[50];
```

```
int Vize;
```

```
int Final;
```

```
};
```

```
int main()
```

```
{
```

```
struct Ogrenci ogrenci_bilgi1,ogrenci_bilgi2;
```

```
printf("*****\n VERI GIRISI\n
```

```
*****\n");
```

```
printf("1. ogrenci bilgileri:\n");
```

```
printf("Ogrenci No:");
```

```
scanf("%d",&ogrenci_bilgi1.No);
```

```
printf("Ogrenci Ad:");
```

```
scanf("%s",ogrenci_bilgi1.Ad);
```

```
printf("Ogrenci Soyad:");
```

```
scanf("%s",ogrenci_bilgi1.Soyad);
```

```
printf("Ogrenci Vize Notu:");
```

```
scanf("%d",&ogrenci_bilgi1.Vize);
```

```
printf("Ogrenci Final Notu:");
```

```
scanf("%d",&ogrenci_bilgi1.Final);
```



```
printf("2. ogrenci bilgileri:\n");
printf("Ogrenci No:");
scanf("%d",&ogrenci_bilgi2.No);
printf("Ogrenci Ad:");
scanf("%s",ogrenci_bilgi2.Ad);
printf("Ogrenci Soyad:");
scanf("%s",ogrenci_bilgi2.Soyad);
printf("Ogrenci Vize Notu:");
scanf("%d",&ogrenci_bilgi2.Vize);
printf("Ogrenci Final Notu:");
scanf("%d",&ogrenci_bilgi2.Final);
```





```
printf("*****\n VERI  
YAZMA\n*****\n");  
printf("1. orenci:\nNo: %d \nAd: %s\nSoyad: %s\nVize: %d\nFinal:  
%d\n",  
ogrenci_bilgi1.No,  
ogrenci_bilgi1.Ad,  
ogrenci_bilgi1.Soyad,  
ogrenci_bilgi1.Vize,  
ogrenci_bilgi1.Final  
);
```

```
printf("2. orenci:\nNo: %d \nAd: %s\nSoyad: %s\nVize: %d\nFinal:  
%d\n",  
ogrenci_bilgi2.No,  
ogrenci_bilgi2.Ad,  
ogrenci_bilgi2.Soyad,  
ogrenci_bilgi2.Vize,  
ogrenci_bilgi2.Final  
);
```

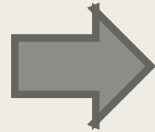
ogrenci_bilgi1=ogrenci_bilgi2;



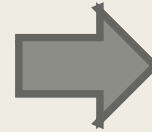
```
printf("*****\n ATAMADAN SONRA VERI  
YAZMA\n*****\n");  
printf("1. orenci:\nNo: %d \nAd: %s\nSoyad: %s\nVize: %d\nFinal:  
%d\n",  
ogrenci_bilgi1.No,  
ogrenci_bilgi1.Ad,  
ogrenci_bilgi1.Soyad,  
ogrenci_bilgi1.Vize,  
ogrenci_bilgi1.Final  
);
```

```
printf("2. orenci:\nNo: %d \nAd: %s\nSoyad: %s\nVize: %d\nFinal:  
%d\n",  
ogrenci_bilgi2.No,  
ogrenci_bilgi2.Ad,  
ogrenci_bilgi2.Soyad,  
ogrenci_bilgi2.Vize,  
ogrenci_bilgi2.Final  
);  
getch ();  
return 0;  
}
```

```
*****
VERI GIRISI
*****
1. ogrenci bilgileri:
Ogrenci No:1234
Ogrenci Ad:ali
Ogrenci Soyad:can
Ogrenci Vize Notu:90
Ogrenci Final Notu:75
2. ogrenci bilgileri:
Ogrenci No:4325
Ogrenci Ad:veli
Ogrenci Soyad:can
Ogrenci Vize Notu:15
Ogrenci Final Notu:85
```



```
*****
VERI YAZMA
*****
1. orenci:
No: 1234
Ad: ali
Soyad: can
Vize: 90
Final: 75
2. orenci:
No: 4325
Ad: veli
Soyad: can
Vize: 15
Final: 85
```



```
*****
ATAMADAN SONRA VERI YAZMA
*****
1. orenci:
No: 4325
Ad: veli
Soyad: can
Vize: 15
Final: 85
2. orenci:
No: 4325
Ad: veli
Soyad: can
Vize: 15
Final: 85
-----
```

struct Değişkenlerin Karşılaştırılması

--- struct tipindeki bir değişken aynı tipteki bir başka struct değişkeni ile direk karşılaştırılamaz.

--- Fakat struct değişkenin sahip olduğu üyeler ile karşılaştırma yapmak mümkündür.

--- Aşağıdaki karşılaştırma yanlıştır:

if (öğrenci.bilgi1 == öğrenci.bilgi2)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct Ogrenci
{
int No;
char Ad[50];
char Soyad[50];
int Vize;
int Final;
} ogrenci_bilgi1,ogrenci_bilgi2;
```

```
int main()
{
ogrenci_bilgi1.No=1123;
ogrenci_bilgi2.No=1129;
printf("1. ogrencinin numarasi= %d\n",ogrenci_bilgi1.No);
printf("2. ogrencinin numarasi= %d\n",ogrenci_bilgi2.No);

if (ogrenci_bilgi1.No==ogrenci_bilgi2.No) printf("Ogrenci Numaralari esittir...\n");
    else printf("Ogrenci Numaralari esit degildir...\n");
getch();
return 0;
}
```

```
1. ogrencinin numarasi= 1123
2. ogrencinin numarasi= 1129
Ogrenci Numaralari esit degildir...
-----
```

struct Değişkenleri ve Fonksiyonlar

--- struct tipindeki bir değişken herhangi bir fonksiyona parametre olarak aktarılabilir.

--- Herhangi bir fonksiyondan geri dönüş değeri olarak struct tipinde bir veri türü geriye dönebilir.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct Ogrenci
{
int No;
char Ad[50];
char Soyad[50];
int Vize;
int Final;
};
void bilgi_yazdir(struct Ogrenci ogr_deneme)
{
printf("\nBilgiler Fonksiyonda Yazdiriliyor:\nNo: %d
\nAd: %s\nSoyad: %s\nVize: %d\nFinal: %d\n",
ogr_deneme.No,
ogr_deneme.Ad,
ogr_deneme.Soyad,
ogr_deneme.Vize,
ogr_deneme.Final
);
}
```

```
int main()
{ struct Ogrenci ogrenci_bilgi1;

printf("Ogrenci bilgilerini giriniz: \n");
printf("Ogrenci No: ");
scanf("%d",&ogrenci_bilgi1.No);
printf("Ogrenci Ad: ");
scanf("%s",ogrenci_bilgi1.Ad);
printf("Ogrenci Soyad: ");
scanf("%s",ogrenci_bilgi1.Soyad);
printf("Ogrenci Vize Notu: ");
scanf("%d",&ogrenci_bilgi1.Vize);
printf("Ogrenci Final Notu: ");
scanf("%d",&ogrenci_bilgi1.Final);
bilgi_yazdir(ogrenci_bilgi1);

getch();
return 0;
}
```

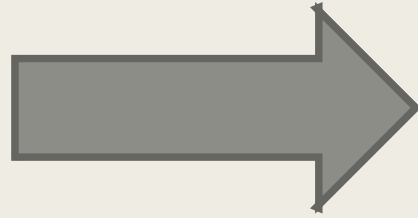

Ekran Çıktısı:

```
Ogrenci bilgilerini giriniz:  
Ogrenci No: 20210019  
Ogrenci Ad: Deniz  
Ogrenci Soyad: İnci  
Ogrenci Vize Notu: 90  
Ogrenci Final Notu: 85  
  
Bilgiler Fonksiyonda Yazdiriliyor:  
No: 20210019  
Ad: Deniz  
Soyad: İnci  
Vize: 90  
Final: 85  
  
-----
```

Herhangi bir fonksiyondan geri dönüş değeri olarak struct tipinde bir veri türü geri döndürmeye örnek:

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

```
struct Ogrenci
{
    int No;
    char Ad[50];
    char Soyad[50];
    int Vize;
    int Final;
};
```



```
struct Ogrenci bilgi_ekle()
{ struct Ogrenci ogrenci_bilgi1;
  printf("Ogrenci bilgileri Fonksiyonda giriliyor: \n");
  printf("Ogrenci No: ");
  scanf("%d",&ogrenci_bilgi1.No);
  printf("Ogrenci Ad: ");
  scanf("%s",ogrenci_bilgi1.Ad);
  printf("Ogrenci Soyad: ");
  scanf("%s",ogrenci_bilgi1.Soyad);
  printf("Ogrenci Vize Notu: ");
  scanf("%d",&ogrenci_bilgi1.Vize);
  printf("Ogrenci Final Notu: ");
  scanf("%d",&ogrenci_bilgi1.Final);
  return ogrenci_bilgi1;
}
```

```
int main()
{ struct Ogrenci ogr_deneme;
  ogr_deneme=bilgi_ekle();
  printf("\nBilgiler Main Fonksiyonunda Yazdiriliyor:\nNo: %d \nAd: %s\nSoyad: %s\nVize: %d\nFinal: %d\n",
  ogr_deneme.No,
  ogr_deneme.Ad,
  ogr_deneme.Soyad,
  ogr_deneme.Vize,
  ogr_deneme.Final
  );

  getch();
  return 0;
}
```

```
Ogrenci bilgileri Fonksiyonda giriliyor:
Ogrenci No: 20210035
Ogrenci Ad: Ali
Ogrenci Soyad: Can
Ogrenci Vize Notu: 90
Ogrenci Final Notu: 88

Bilgiler Main Fonksiyonunda Yazdiriliyor:
No: 20210035
Ad: Ali
Soyad: Can
Vize: 90
Final: 88
-----
```

struct Değişkenler ve Diziler

- struct tipi içerisinde, dizi türünde üyeler tanımlamak mümkündür.
- Dizileri, struct tipinde tanımlamak mümkündür.

Nesneye dayalı programlama yaklaşımı için önemli!!!

Örnek1: ogrnot isimli bir struct aşağıdaki üyeleri içersin:

- Öğrenci Numarasını,
- 2 tane Vize Notunu,
- 3 tane Ödev Notunu,
- 1 tane Final Notunu.



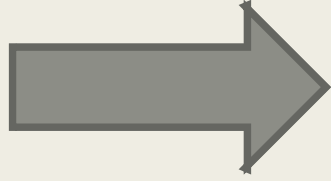
```
struct ogrnot
{
    int no;
    int vize[2]; // struct içinde dizi tanımlandı...
    int odev[3]; // struct içinde dizi tanımlandı...
    int final;
};
```

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

```
struct ogrnot
{
int no;
int vize[2]; // struct içinde dizi tanımlandı...
int odev[3]; // struct içinde dizi tanımlandı...
int final;
};
```

EKRAN ÇIKTISI

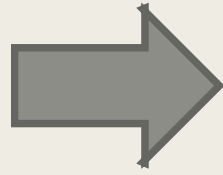
```
Ogrenci bilgileri yazdiriliyor:
No: 20210011
1.vize: 78
2.vize: 90
1.odev: 100
2.odev: 95
3.odev: 98
final: 80
-----
```



```
int main()
{ struct ogrnot ogr;
ogr.no=20210011;
ogr.vize[0]=78;
ogr.vize[1]=90;
ogr.odev[0]=100;
ogr.odev[1]=95;
ogr.odev[2]=98;
ogr.final=80;
printf("Ogrenci bilgileri yazdiriliyor:\nNo: %d
\n1.vize: %d \n2.vize: %d \n1.odev: %d \n2.odev: %d
\n3.odev: %d \nfinal: %d", ogr.no, ogr.vize[0],
ogr.vize[1],
ogr.odev[0],
ogr.odev[1],
ogr.odev[2],
ogr.final
);
getch();
return 0;
}
```

Örnek1: ogrnot isimli bir struct aşağıdaki üyeleri içersin ve 3 öğrenci için tanımlansın:

--- Öğrenci Numarasını,
--- 2 tane Vize Notunu,
--- 3 tane Ödev Notunu,
--- 1 tane Final Notunu.



```
struct ogrnot
{
    int no;
    int vize[2]; // struct içinde dizi tanımlandı...
    int odev[3]; // struct içinde dizi tanımlandı...
    int final;
};
```

```
int main()
{ struct ogrnot ogr[3]; // 3 elemanlı dizi, elemalar struct

}
```

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct ogrnot
{
int no;
int vize[2]; // struct içinde dizi tanımlandı...
int odev[3]; // struct içinde dizi tanımlandı...
int final;
};
int main()
{ struct ogrnot ogr[3];

ogr[0].no=20210011;
ogr[0].vize[0]=78;
ogr[0].vize[1]=90;
ogr[0].odev[0]=100;
ogr[0].odev[1]=95;
ogr[0].odev[2]=98;
ogr[0].final=80;
```



```
ogr[1].no=20210016;
ogr[1].vize[0]=45;
ogr[1].vize[1]=55;
ogr[1].odev[0]=90;
ogr[1].odev[1]=100;
ogr[1].odev[2]=60;
ogr[1].final=45;
```

```
ogr[2].no=20210036;
ogr[2].vize[0]=15;
ogr[2].vize[1]=65;
ogr[2].odev[0]=95;
ogr[2].odev[1]=70;
ogr[2].odev[2]=55;
ogr[2].final=95;
```





```
printf("1. Ogrenci Bilgileri:\nNo: %d \n1.vize: %d \n2.vize: %d\n1.odev: %d \n2.odev: %d \n3.odev: %d \nfinal: %d",  
ogr[0].no,  
ogr[0].vize[0],  
ogr[0].vize[1],  
ogr[0].odev[0],  
ogr[0].odev[1],  
ogr[0].odev[2],  
ogr[0].final  
);
```

```
printf("\n2. Ogrenci Bilgileri:\nNo: %d \n1.vize: %d \n2.vize: %d \n1.odev: %d \n2.odev: %d \n3.odev: %d \nfinal: %d",  
ogr[1].no,  
ogr[1].vize[0],  
ogr[1].vize[1],  
ogr[1].odev[0],  
ogr[1].odev[1],  
ogr[1].odev[2],  
ogr[1].final  
);
```





EKRAN ÇIKTISI

```
printf("\n3. Ogrenci Bilgileri:\nNo: %d \n1.vize: %d\n2.vize: %d \n1.odev: %d \n2.odev: %d \n3.odev: %d\nfinal: %d",  
ogr[2].no,  
ogr[2].vize[0],  
ogr[2].vize[1],  
ogr[2].odev[0],  
ogr[2].odev[1],  
ogr[2].odev[2],  
ogr[2].final  
);  
  
getch();  
return 0;  
}
```

```
1. Ogrenci Bilgileri:  
No: 20210011  
1.vize: 78  
2.vize: 90  
1.odev: 100  
2.odev: 95  
3.odev: 98  
final: 80  
2. Ogrenci Bilgileri:  
No: 20210016  
1.vize: 45  
2.vize: 55  
1.odev: 90  
2.odev: 100  
3.odev: 60  
final: 45  
3. Ogrenci Bilgileri:  
No: 20210036  
1.vize: 15  
2.vize: 65  
1.odev: 95  
2.odev: 70  
3.odev: 55  
final: 95  
-----
```

typedef Kullanımı

- typedef komutu C dilinde değişken tanımlama yaparken kullanılan int, float, char gibi değişken isimlerini değiştirmeye yarar.
- Programcıya kolaylık sağlar.
- struct yapısı ile beraber kullanıldığında oluşturduğunuz yapıyı bir değişken türü olarak tanımlayıp o yapının çoğaltılmasını sağlar.
- Böylece, değişken tanımlamak için tekrar struct deyiminin kullanılmasına gerek kalmamaktadır.

```
typedef struct
```

```
{  
int no;  
int vize;  
int odev;  
int final;  
} ogrnot;
```

```
int main()
```

```
{  ogrnot ogr; // struct yazmadan tanımladık...
```

```
}
```

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

```
typedef struct
{
int no;
int vize;
int odev;
int final;
} ogrnot;
```

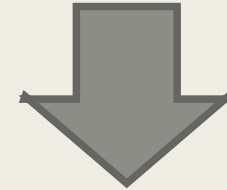
```
int main()
{  ogrnot ogr; // struct yazmadan tanımladık...
```

```
ogr.no=20210011;
ogr.vize=78;
ogr.odev=100;
ogr.final=80;
```



```
printf("Ogrenci Bilgileri:\nNo: %d
\nvize: %d \nodev: %d \nfinal: %d",
ogr.no,
ogr.vize,
ogr.odev,
ogr.final
);

getch();
return 0;
}
```



```
Ogrenci Bilgileri:
No: 20210011
vize: 78
odev: 100
final: 80
-----
```

struct içerisinde struct Kullanımı

- C programlama dilinde struct içerisindeki bir üye yine bir struct tipinde olabilir.
- Kaynak kodu sadeleştirmek, kaynak kodu okunabilir yapmak bu işlemi yapmaktaki temel amaçtır.

Nesneye dayalı programlama yaklaşımı için önemli!!!

Örnek:

- Vize, final ve 2 ödev bilgisi saklayan ogrenci_not isimli bir struct oluşturun.
- e-posta adresi ve telefon numarası bilgilerini saklayan ogrenci_iletisim isimli bir struct oluşturun.
- Öğrenci no, ad, soyad, ogrenci_not üyeleri ve ogrenci_iletisim üyelerini saklayan ogrenci isimli bir struct oluşturunuz...

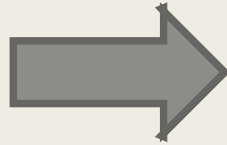
struct içinde struct tanımlanacaktır!!!

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

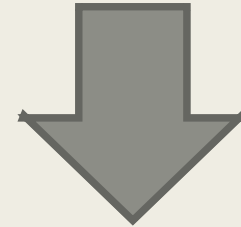
```
typedef struct
{
int vize;
int odev[2];
int final;
} ogrenci_not;
```

```
typedef struct
{
char ceptel[20];
char eposta[20];
} ogrenci_iletisim;
```

struct içinde struct tanımlandı...

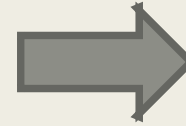


```
typedef struct
{
int no;
char ad[20];
char soyad[20];
ogrenci_not ogrnot;
ogrenci_iletisim ogriletisim;
} ogrenci;
```





```
int main()
{   ogrenci ogr;
    ogr.no=20210011;
    strcpy(ogr.ad,"ali");
    strcpy(ogr.soyad,"can");
    strcpy(ogr.ogriletisim.ceptel,"05851110011");
    strcpy(ogr.ogriletisim.eposta,"alican@xxxxxxx.com");
    ogr.ogrnot.vize=75;
    ogr.ogrnot.final=95;
    ogr.ogrnot.odev[0]=100;
    ogr.ogrnot.odev[1]=90;
```



```
printf("Ogrenci Bilgileri:\nNo: %d \nad:
%s \nsoyad: %s \ntelefon: %s \ne-posta:
%s \nvize: %d \nfinal: %d  \n1.odev:
%d  \n2.odev: %d",
ogr.no,
ogr.ad,
ogr.soyad,
ogr.ogriletisim.ceptel,
ogr.ogriletisim.eposta,
ogr.ogrnot.vize,
ogr.ogrnot.final,
ogr.ogrnot.odev[0],
ogr.ogrnot.odev[1]
);

getch();
return 0;
}
```

```
Ogrenci Bilgileri:
No: 20210011
ad: ali
soyad: can
telefon: 05851110011
e-posta: alican@xxxxxxx.com
vize: 75
final: 95
1.odev: 100
2.odev: 90
-----
```

Dosyalar (Files)

Dosyalara Giriş

- Şu anda gördüğümüz değişkenlerde ve dizilerde depolanan veriler bellekte tutulurlar ve geçicidirler.
- Bu veriler program sonlandığında bellekten kaybolurlar.
- Dosyalar ise büyük miktarda veriyi kalıcı olarak tutmak için kullanılmaktadır.
- Bilgisayarlar dosyaları genellikle ikincil depolama cihazlarında tutmaktadır.
- Disk depolama cihazları buna örnek verilebilir.

- C programlama dili, her bir dosyayı bitlerin art arda geldiği bir akış olarak görür.
- Her dosya, **ya** dosya sonu belirteci (end-of-file) **ya da** sistemde yönetici veri yapısı tarafından belirlenmiş özel bir byte sayısı ile sonlanmaktadır.
- Akışlar, dosyalar ile program arasında haberleşme kanalları oluşturur. Örneğin, standart giriş akışı programın klavyeden veri okumasını ve standart çıkış akışı programın ekrana veri yazdırmasını sağlamaktadır.

Genel Dosya Komutları

--- Bir dosyaya okuma ve yazma yapmak için yapılması gereken ilk işlem dosyayı açmaktır.

--- Dosyayı açmak için **fopen()** komutu kullanılır.

--- Dosyayı kapatmak için **fclose()** komutu kullanılır.

--- Bu fonksiyonlar `<stdio.h>` kütüphanesi içerisinde yer almaktadır.

--- C dilinde dosya açmak ve kapatmak aşağıdaki şekildedir:

```
FILE *pDosya; /* dosya işaretçisi*/  
pDosya = fopen(const char dosya_adı, const char mod);  
...  
...  
...  
fclose(pDosya);
```

--- *pDosya ifadesi, pDosya'nın FILE yapısını gösteren bir işaretçi olduğunu belirtmektedir.

--- C programı, her dosyayı ayrı bir FILE yapısıyla yönetir.

--- Programı yazan yazılımcı dosyaları kullanabilmek için FILE yapısının özelliklerini mutlaka bilmelidir.

--- Dosya açma işlemi için kullanılan **fopen()** fonksiyonundaki **mod** değişkeni için kullanılan değişkenler ve anlamları aşağıdaki gibidir:

r : Sadece okuma amaçlı dosyayı açmak için kullanılır. Dosya daha önceden oluşturulmuş olmalıdır.

w: Sadece yazma amaçlı dosyayı açmak için kullanılır. Dosya var olsun ya da olmasın yeniden oluşturulur. Dosya varsa eski bilgiler silinir. Bu modda açılan dosyadan okuma yapılamaz.

a : Sadece dosya sonuna eklemek yapmak amaçlı dosyayı açmak için kullanılır. Bu modda açılan dosyadan okuma yapılamaz.

r+: Okuma ve yazma amaçlı dosyayı açmak için kullanılır. Dosya daha önceden dosya oluşturulmuş olmalıdır.

w+: Okuma ve yazma amaçlı dosyayı açmak için kullanılır. Dosya var olsun ya da olmasın yeniden oluşturulur.

a+: Okuma ve yazma amaçlı dosyayı açmak için kullanılır. Kayıtlı bir dosyanın sonuna veri eklemek için kullanılır.

--- Bir dosyaya erişmek ve üzerinde işlem yapabilmek için, dosyanın açılıp açılmadığını test etmemiz gerekir.

--- Aşağıda bir dosya açmaya, dosyanın açılıp açılmadığını test etmeye ve dosyayı kapatmaya bir örnek verilmiştir.

Örnek: FILE *cfptr;

```
if ( ( cfptr = fopen("ogrenci.txt", "w" ) ) == NULL )  
    printf( "Dosya acilmadi\n" );  
else {  
    .  
    .  
    .  
    fclose( cfPtr );  
}
```

fprintf() ve fscanf() fonksiyonları:

- **fprintf()** fonksiyonu, dosyaya veri yazmak için kullanılır.
- Eğer dosyaya bir sayı (int) yazılacaksa, sayının dosyanın ASCII metnine çevrilmesi gerekir.
- Kullanımı: `fprintf(cfptr, "%d", sayi);` şeklindedir.

- **fscanf()** fonksiyonu, dosyadan veri okumak için kullanılır.
- Eğer dosyadan bir sayı (int) okunacaksa, sayının `fscanf()` fonksiyonunun dahili format yapısına çevrilmesi gerekir.
- Kullanımı: `fscanf (fptr, "%d", &sayi);` şeklindedir.

Örnek 1: Belirsiz sayıda müşterinin, müşteri numarasını, ismini ve bakiye miktarını "musteri.txt" isimli dosyaya yazdıran bir C programı yazınız.

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

```
int main()
{
    int num;
    char isim[ 30 ];
    int bakiye;
    FILE *cfptr;

    if ( ( cfptr = fopen("musteri.txt", "w" ) ) == NULL )
        printf( "Dosya acilmadi\n" );
    else {
        printf( "Musteri numarasini giriniz: ");
        scanf( "%d", &num );
        printf( "Musterinin ismini giriniz: ");
        scanf( "%s", isim);
        printf( "Musteri bakiyesini giriniz: ");
        scanf( "%d", &bakiye );
```



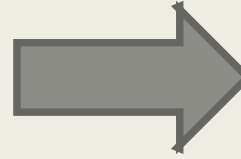
```
while ( !feof( stdin ) ) {
    fprintf( cfptr, "%d %s %d\n", num, isim, bakiye );
    printf( "EOF girerek veri girisini sonlandiriniz.\n" );
    printf( "Musteri numarasini giriniz: ");
    scanf( "%d", &num );
    printf( "Musterinin ismini giriniz: ");
    scanf( "%s", isim);
    printf( "Musteri bakiyesini giriniz: ");
    scanf( "%d", &bakiye );}

    fclose( cfptr );
}

getch();
return 0;
}
```



```
Musteri numarasini giriniz: 3219
Musterinin ismini giriniz: ali
Musteri bakiyesini giriniz: 9500
EOF girerek veri girisini sonlandiriniz.
Musteri numarasini giriniz: 5430
Musterinin ismini giriniz: veli
Musteri bakiyesini giriniz: 800
EOF girerek veri girisini sonlandiriniz.
Musteri numarasini giriniz: ^Z
Musterinin ismini giriniz: ^Z
Musteri bakiyesini giriniz: ^Z
-----
```



musteri - Not Defteri

Dosya	Düzen	Biçim	Görünüm	Yardım
-------	-------	-------	---------	--------

3219	ali	9500		
------	-----	------	--	--

5430	veli	800		
------	------	-----	--	--

- **stdin**, C programlama dilinde bir dosya tanımlayıcısıdır ve verinin bir programa gönderildiği ve bir program tarafından okunduğu bir giriş akışı olarak ifade edilir.
- EOF (ctrl+Z) girerek veri girişi sonlandırılır.

Örnek 2: Örnek 1 de oluşturulan dosyanın verilerini dosyadan okuyup ekrana yazdıran bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int num;
  char isim[ 30 ];
  int bakiye;
  FILE *cfptr;

  if ( ( cfptr = fopen("musteri.txt", "r" ) ) == NULL )
    printf( "Dosya acilamadi\n" );
  else {
    printf( "%-15s%-10s%s\n", "Musteri No", "Isim", "Bakiye" );
    fscanf( cfptr, "%d%s%d", &num, isim, &bakiye );

    while ( !feof( cfptr ) ) {
      printf( "%d %13s %11d\n", num, isim, bakiye );
      fscanf( cfptr, "%d%s%d", &num, isim, &bakiye ); }
    fclose( cfptr );
  }
  getch();
  return 0;
}
```

EKRAN ÇIKTISI

Musteri No	Isim	Bakiye
3219	ali	9500
5430	veli	800

```
if ( ( cfptr = fopen("muste.txt", "r" ) ) == NULL )
  printf( "Dosya acilamadi\n" );
else { ...
```

şeklinde yazılırsa program aşağıdaki çıktıyı verir:

```
Dosya acilamadi
-----
Process exited after 12 seconds with return value 0
Press any key to continue . . . _
```

"muste.txt" isimli bir dosya yok!!!

Örnek 3: Örnek 1 de oluşturulan musterit.txt isimli dosyaya yeni bir kayıt olacak şekilde dosyayı açan ve daha sonra dosyanın son halini listeleyen bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int num;
  char isim[ 30 ];
  int bakiye;
  FILE *cfptr;


  if ( ( cfptr = fopen("musterit.txt", "a" ) ) == NULL )
    printf( "Dosya acilmadi\n" );
  else {
    printf( "Yeni musterinin numarasini giriniz: ");
    scanf( "%d", &num );
    printf( "Yeni musterinin ismini giriniz: ");
    scanf( "%s", isim);
    printf( "Yeni musterinin bakiyesini giriniz: ");
    scanf( "%d", &bakiye );
    fprintf( cfptr, "%d %s %d\n", num, isim, bakiye );
    fclose( cfptr ); }
```



```
if ( ( cfptr = fopen("musterit.txt", "r" ) ) == NULL )
  printf( "Dosya acilamadi\n" );
else {
  printf( "%-15s%-10s%s\n", "Musteri No", "Isim",
"Bakiye" );
  fscanf( cfptr, "%d%s%d", &num, isim, &bakiye );

  while ( !feof( cfptr ) ) {
    printf( "%d %13s %11d\n", num, isim, bakiye );
    fscanf( cfptr, "%d%s%d", &num, isim,
&bakiye ); }
  fclose( cfptr );
  }
  getch();
  return 0;
}
```



 musteriler - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

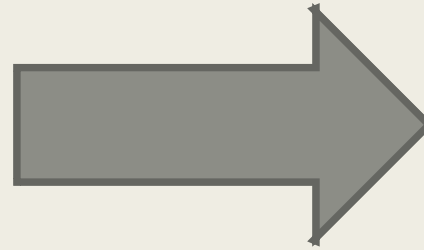
3219	ali	9500
5430	veli	800
2367	berk	1650



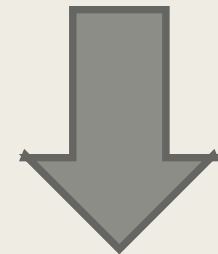
```
Yeni musterinin numarasini giriniz: 2367
Yeni musterinin ismini giriniz: berk
Yeni musterinin bakiyesini giriniz: 1650
Musteri No      Isim      Bakiye
3219            ali      9500
5430            veli      800
2367            berk      1650
-----
```

Örnek 4: Öğrenci numarası, ad, soyad, vize ve final notlarını içeren bir struct oluşturup, bu structaki bilgileri deneme.txt isimli dosyaya yazdıran bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct Ogrenci
{
int No;
char Ad[50];
char Soyad[50];
int Vize;
int Final;
};
int main()
{ FILE *cfptr;
  struct Ogrenci ogrenci_bilgi;
  if ( ( cfptr = fopen("deneme.txt", "w" ) ) == NULL )
    printf( "Dosya acilamadi\n" );
```



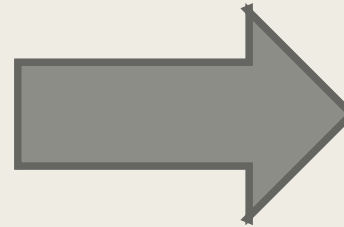
```
else {
    printf("Ogrenci No:");
    scanf("%d",&ogrenci_bilgi.No);
    printf("Ogrenci Ad:");
    scanf("%s",ogrenci_bilgi.Ad);
    printf("Ogrenci Soyad:");
    scanf("%s",ogrenci_bilgi.Soyad);
    printf("Ogrenci Vize Notu:");
    scanf("%d",&ogrenci_bilgi.Vize);
    printf("Ogrenci Final Notu:");
    scanf("%d",&ogrenci_bilgi.Final);
```





```
while ( !feof( stdin ) ) {  
    fprintf( cfptr, "%d %s %s %d %d\n",  
ogrenci_bilgi.No, ogrenci_bilgi.Ad, ogrenci_bilgi.Soyad,  
ogrenci_bilgi.Vize, ogrenci_bilgi.Final);  
    printf( "EOF girerek veri girisini sonlandiriniz.\n" );  
    printf("Ogrenci No:");  
    scanf("%d",&ogrenci_bilgi.No);  
    printf("Ogrenci Ad:");  
    scanf("%s",ogrenci_bilgi.Ad);  
    printf("Ogrenci Soyad:");  
    scanf("%s",ogrenci_bilgi.Soyad);  
    printf("Ogrenci Vize Notu:");  
    scanf("%d",&ogrenci_bilgi.Vize);  
    printf("Ogrenci Final Notu:");  
    scanf("%d",&ogrenci_bilgi.Final);}  
    fclose( cfptr );  
}  
getch ();  
return 0;  
}
```

```
Ogrenci No:1234
Ogrenci Ad:can
Ogrenci Soyad:berk
Ogrenci Vize Notu:80
Ogrenci Final Notu:95
EOF girerek veri girisini sonlandiriniz.
Ogrenci No:2345
Ogrenci Ad:veli
Ogrenci Soyad:ali
Ogrenci Vize Notu:85
Ogrenci Final Notu:100
EOF girerek veri girisini sonlandiriniz.
Ogrenci No:1256
Ogrenci Ad:burak
Ogrenci Soyad:cemal
Ogrenci Vize Notu:45
Ogrenci Final Notu:88
EOF girerek veri girisini sonlandiriniz.
Ogrenci No:^Z
Ogrenci Ad:^Z
Ogrenci Soyad:^Z
Ogrenci Vize Notu:^Z
Ogrenci Final Notu:^Z
-----
```



deneme - Not Defteri

Dosya Düzen Biçim Görünüm Yardım


1234 can berk 80 95

2345 veli ali 85 100

1256 burak cemal 45 88

Örnek 5: Örnek 4 de oluşturulan deneme.txt isimli dosyanın verilerin dosyadan okuyup ekrana yazdıran bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int num;
  char ad[ 30 ];
  char soyad[ 30 ];
  int vize;
  int final;
  FILE *cfptr;
  if ( ( cfptr = fopen("deneme.txt", "r" ) ) == NULL )
    printf( "Dosya acilamadi\n" );
  else {
    printf( "%-15s%-11s%-15s%-12s%s\n", "Ogrenci No", "Isim","Soyisim", "Vize", "Final" );
    fscanf( cfptr, "%d%s%s%d%d", &num, ad, soyad, &vize, &final );
    while ( !feof( cfptr ) ) {
      printf( "%d %13s %13s %11d %11d\n", num, ad, soyad, vize, final);
      fscanf( cfptr, "%d%s%s%d%d", &num, ad, soyad, &vize, &final ); }
    fclose( cfptr );
  }
  getch();
  return 0;
}
```

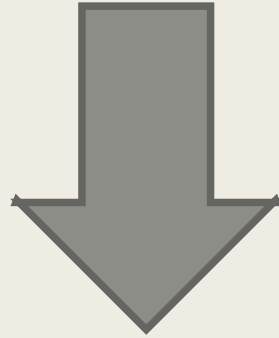
 deneme - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

1234 can berk 80 95

2345 veli ali 85 100

1256 burak cemal 45 88



Ogrenci No	Isim	Soyisim	Vize	Final
1234	can	berk	80	95
2345	veli	ali	85	100
1256	burak	cemal	45	88

Process exited after 4.586 seconds with return value 0

Kaynaklar

- C: How to Program Third Edition Harvey M. Deitel ; Paul J. Deitel.
- C Programlama Dili Dr. Rıfat Çölkesen Papatya Yayıncılık.
- Problem Solving and Program Design in C, 7/E Jeri R. Hanly; Elliot B. Koffman.
- C Programlama dili; İbrahim Güney; Nobel Yayıncılık.
- Algoritma Geliştirme ve Programlamaya Giriş, Fahri Vatansever, Seçkin yayıncılık
- C Programlama Ders Notları, A. Kadir YALDIR, Pamukkale Üniversitesi ders notları.