T.C
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

ENDÜSTRIYEL OTOMASYON TEKNOLOJILERI

TEMEL PROGRAMLAMA- 3

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**523EO0357**
ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Temel Programlama 3**
MODÜLÜN TANIMI	Bilgisayar programı yazım tekniklerini anlatan öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Temel Programlama 2 modülünü almış olmak
YETERLİK	Bilgisayar programı yazmak
MODÜLÜN AMACI	**Genel Amaç** Bilgisayar programını programlama dili tekniklerine uygun olarak yazabileceksiniz. **Amaçlar** 1. Bilgisayar programlamada dizi yapısındaki verileri doğru bir şekilde kullanabileceksiniz. 2. Bilgisayar programlamada karakter ve karakter katarlarını doğru bir şekilde kullanabileceksiniz. 3. Bilgisayar programlamada disk dosyalarının yazma okuma ve ekleme işlemlerini doğru bir şekilde yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Bilgisayar Laboratuvarı **Donanım:** Bilgisayar, bilgisayar çevre birimleri, programlama akış şablonu
ÖLÇME VE DEĞERLENDİRME	Ø Modülün içinde yer alan her öğrenme faaliyetinden sonra, verilen performans testi ile kendinize ilişkin gözlem ve değerlendirmeleriniz yoluyla kazandığınız bilgi ve becerileri ölçerek kendinizi değerlendirebileceksiniz. Ø Öğretmen, modül sonunda size ölçme teknikleri uygulayarak modül uygulamalarıyla kazandığınız bilgi ve becerileri ölçerek değerlendirebilecektir.

GİRİŞ

Sevgili Öğrenci,

Bundan önceki modülde C dilinin temelleri üzerinde çalıştınız. Bu bilgiler vazgeçilmez olgulardır ve sınırlı kullanıma sahip programlar için yeterlidir. Başka bir ifadeyle bu temel bilgiler fazla veriye sahip daha büyük programlar için yeterli değildir. Program dışında veri taşıyabilen yardımcı dosyalar ve program yapısını oluşturma imkanı veren fonksiyonlarla, daha karışık ve uygulamaya yönelik programlar yazabiliriz. Bu nedenle modülde sistem gelişim süreci ile ilişkili olan bu konular üzerinde çalışacaksınız.

Programlar 50 satırı aştığı zaman, iyi bir yapıyla net bir şekilde düzenlenmeleri zorunlu hale gelir. Aksi takdirde programın ayıklanması, çok fazla zaman alacak ve programcı sonradan bazı bölümleri değiştirmek istediği zaman, çok fazla zorluklarla yüz yüze gelecektir. Bu bilgiler bireysel tecrübelerden elde edilmiştir. Bu modülde kendinizi, kendi fikirlerinizden yola çıkarak büyük programlar yapmaya teşvik etmelisiniz.

ÖĞRENME FAALİYETİ–1

AMAÇ

Bilgisayar programlamada dizi yapısındaki verileri doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

Bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapmalısınız.

Ø C programlamada dizi yapıları hakkında araştırma yapınız.

1. DİZİLER

C programlama dilinde veriler değişken yapısı içinde saklanır. Bir değişken sadece bir veriyi tutar. Programda birçok veriyi kullanmak istendiğinde ise **dizi** adı verilen veri yapıları kullanılır. Dizi, belirli sayıda verinin bellekte saklandığı değişken listeleridir.

1.1. Tek Boyutlu Diziler

Örnek 1.1.

4 adet integer tipindeki sayıyı tutan ve bunları giriş sırasına göre görüntüleyen programı yazınız.

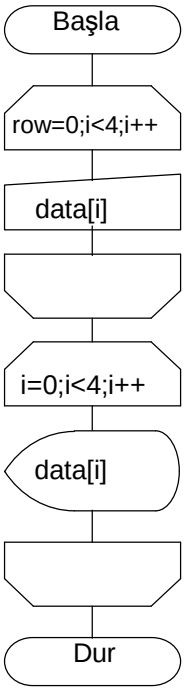
Ø **Ekran Görüntüsü**

```
Sayı Giriniz =>25
Sayı Giriniz  =>48
Sayı Giriniz  =>17
Sayı Giriniz  =>3
25
48
17
3
```

Ø Değişken Tablosu

Değişken	tip	Kullanım
data[4]	int	Saklanan sayı
i	int	Kontrol değişkeni

Ø Akış Diyagramı




```
/* Örnek 1.1 */

#include <stdio.h>

main()
{
     int data[4],i;                                  //--1

    for(i=0;i<4;i++)
   {                                                 //--2
        printf("Sayı giriniz=>");
        scanf("%d",&data[i]);
    }
   for(i=0;i<4;i++)                                  //--3
        printf("%d \n",data[i]);
}
```

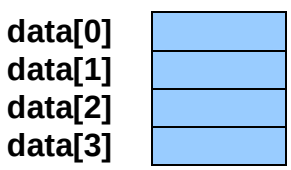
Ø Program Açıklaması

1- **int data[4];** komutuyla dizinin tanımlanması.

2- 1. maddede tanımlanan dizi için klavyeden bilgi girişinin yapılması

3- Dizinin içeriğinin görüntülenmesi

int data[4] şeklindeki bir tanımlamayla, şekilde görüldüğü gibi, ana bellekte sıralı olarak hafıza alanı hazırlanır. Bu yapıya dizi adı verilir. Dizilerin kullanım yapısı şu şekildedir.



Dizinin her elemanına bir dizi ismi ve köşeli parantezlerin içerisine yazılan eleman numarası ile ulaşabiliriz. Dizilerde köşeli parantezler (subscript) için 0 dan başlama kuralı olduğundan, data[4] şeklindeki bir tanımlamada, 4 eleman data[0], data[1], data[2], data[3] ü ifade eder.

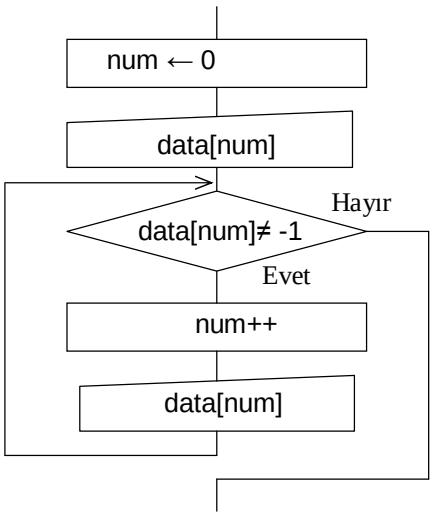
Örneğin;data[1]=5 yazdığımızda, dizinin ikinci elemanına sayısal 5 değerini aktarıyoruz demektir.

Aynı zamanda, köşeli parantezler içerisine değişken ismini de yazabiliriz.

```
i=1;
data[i]=5;
```

Yukarıdaki ifade bir öncekiyle aynıdır. Bu yazım şeklinin döngü içinde kullanılması diziler üzerinde işlem yapabilmemizi sağlar.

Örnek 1.1 de , veri sayısı 4 ile sınırlıdır. Programı istediğimiz kadar veriyi kullanabilmek için değiştirebiliriz. Bunu gerçekleştirebilmek için gerekli sayıda eleman kullanan bir dizi aşağıdaki örnekte tanıtılmıştır.



Bu programda dizi sonu - 1 değerinin girilmesiyle kontrol edilmektedir.
Verilerin girilmesinden sonra num değişkeni veri sayısını tutmaktadır.

Şekil 1.1: Karar yapılı döngü kontrolü

Örnek 1.2

İsteğe bağlı olarak 10`a kadar sayı girilebilen ve menüde gösterilen işlemleri gerçekleştirecek programı yazınız. Veri ekleme ve sıralama daha sonra anlatılacaktır.

Ø **Değişken Tablosu**

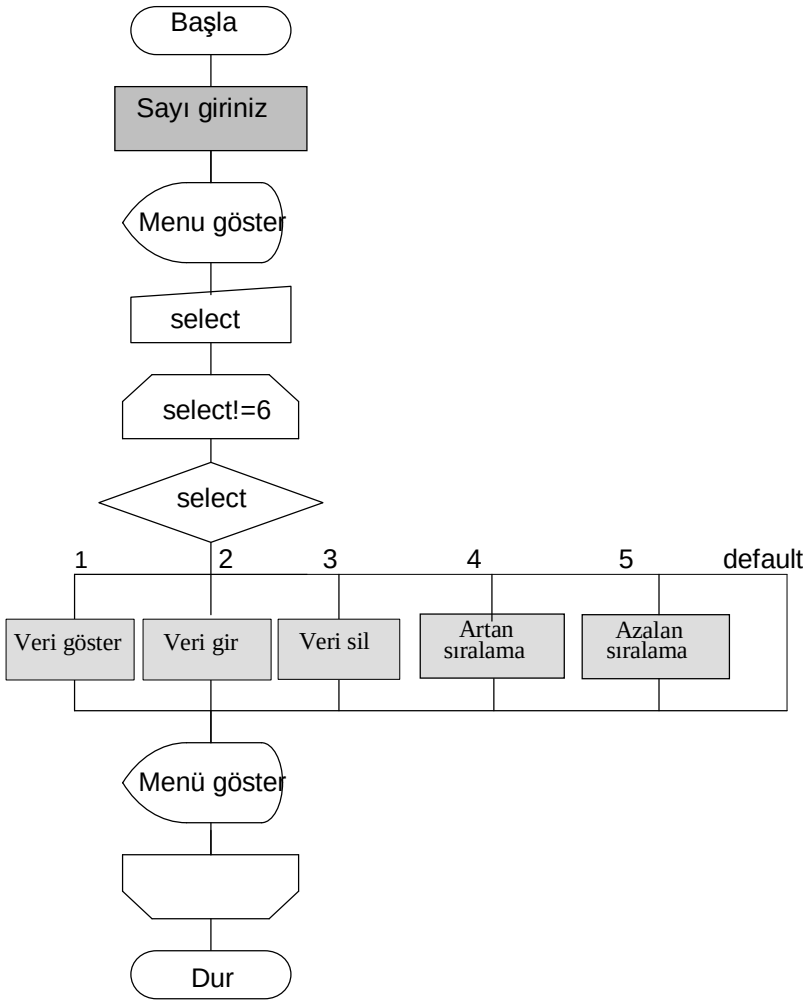
değişken	tip	Kullanım
data[11]	int	Sayıyı saklama
select	int	Menu seçimi
num	int	Veri sayısı
i	int	Kontrol değişkeni

Ø **Ekran Görüntüsü**

```
Sayı giriniz(-1 Bitir)==> 20
Sayı giriniz (-1 Bitir)==> 10
Sayı giriniz (-1 Bitir)==> 30
Sayı giriniz (-1 Bitir)==> -1

Menu
1 Veri göster
2 Veri sil
3 Veri gir
4 Artan sıralama
5 Azalan sıralama
6 bitir
Seçiminiz ==>
```

Bu programda karmaşıklığa sebep olmamak için birçok printf komutu menu fonksiyonunda kullanılmıştır. Buna göre programa veri girildiğinde menu fonksiyonu işletilecektir. Bu fonksiyonu bir alt program olarak düşünebiliriz. Başka bir ifadeyle programın herhangi bir yerinde **menu();** yazarak printf komutlarının grup halinde yazıldığı menu fonksiyonunu çağırabiliriz. C programlamada fonksiyon isimlerine programcı karar verir.

Ø **Akış Diyagramı**



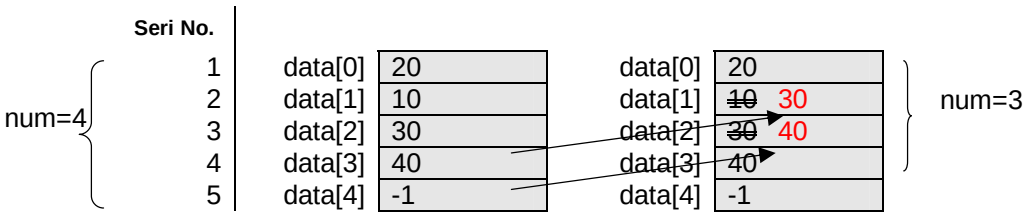
Burada genel bir akış diyagramı program yapısının tamamını göstermek için kullanılmaktadır. Kullanılan kutular tek bir süreci değil bir gurup çalışmayı gösterir. Bu şekilde büyük programlar yaptığınızda, öncelikle programın genel yapısını tasarlayıp sonra detaya girmek istenen bir durumdur.

Ø **Program**

```
/* Ornek 1.2 */

#include<stdio.h>
main()
{
   int data[11],select,num,i;

   num = 0;
   printf("Sayi giriniz(-1 bitir)==>");
   scanf("%d",&data[num]);
   while (data[num] != -1){
         num++;
         printf("Sayi giriniz(-1 bitir)==>");
         scanf("%d",&data[num]);
   }
   menu();
   scanf("%d",&select);
   while( select != 6) {
       switch (select){
          case 1:
               printf("\nNo.  Number\n");
                i=0;
               while(i<num) {
                   printf("%2d %6d\n",i+1,data[i]);
                   i++;
               }
               break;
           case 2:
     printf("Silmek istediginiz numarayi giriniz ==> ");
               scanf("\d",&i);
               while (i < num){
                   data[i-1]=data[i];
                   i++;
               }
                   num--;
               break;
          case 3:
              /*  veri girişi için program */
               break;
          case 4:
             /*  artan siralama icin program */
               break;
          case 5:
              /* azalan siralama için program */
               break;
       }
       menu();
       scanf("%d",&select);
   }
}
menu()
{
   printf("\nMenu\n");
   printf("1 – Veri goster\n");
   printf("2 – Veri sil\n");
   printf("3 – Veri gir\n");
   printf("4 – Artan siralama\n");
   printf("5 – Azalan siralama\n");
   printf("6 -- Bitir\n");
   printf("Seciminiz ==> ");
}
```

Ø Program Açıklaması

Verinin şekilde görüldüğü gibi dizi içindeki yerinin değiştirilmesi ile bu işlem gerçekleştirilir. Aşağıdaki şekilde dizi içindeki 10 sayısı siliniyor.



Şekil 1.2: Silme işlemi

Yukarıda görülen şekli kullanarak , sürecin taslağını gösteren aşağıdaki gibi bir akış diyagramı çizebilmeliyiz. Bunun için (1), (2) ve (3) numaralı adımlara aşağıdaki gibi karar verilmiştir

Şekil 1.2 yi göz önüne alırsak verileri şu şekilde taşıdık,

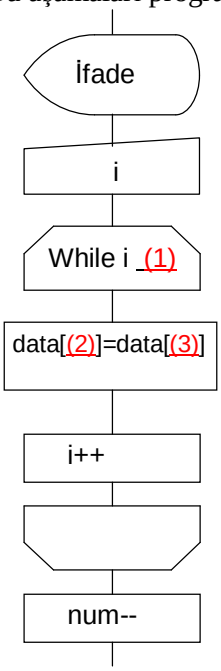
data[1] ← data[2],

data[2] ← data[3] ve i değeri 2 den başladı. (2) ve (3) e şu şekilde karar verebiliriz.

data[i-1]= data[i].

(2) (3)

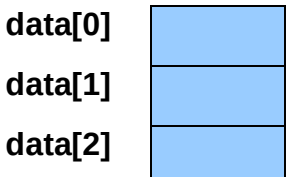
Benzer şekilde, sayıları karşılaştırarak (1) değerini **< num** şeklinde sonlandırabiliriz. -1 değerinin data[4] ten data[3] e taşınmasına ihtiyaç duymadığımız için, num değerinden 1 sayısını çıkarmak zorundayız.Bu aşamaları program takibinde dikkate almalısınız!



1.1.1. Dizinin Başlatılması

Bir dizinin başlatılması işlemi şu şekilde tanımlanabilir

int data[4] = { 456, 23, 567, 89};

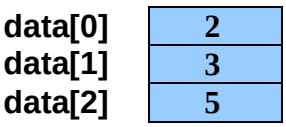


Bu işlem dizinin tanımlanması ve ilk değerinin verilmesi ile tamamlanır.Dizinin başlatılması bir atama olmadığından dolayı aşağıdaki şekilde yazamayız.

```
int data[4];
data[4] = { 456, 23, 567, 89 };
```

Bazen başlatma yapılırken dizinin boyutunu ihmal edebiliriz.

int data[ ] = { 2,3,5 };.



Bu durumda dizinin boyutu yukarıdaki şekilde olduğu gibi otomatik olarak belirlenen veri sayısınca ayarlanır.

1.1.2. Sıralama

Örnek 1.3.

Bir dizideki sayıları azalan şekilde sıralayan ve görüntüleyen programı yazınız.

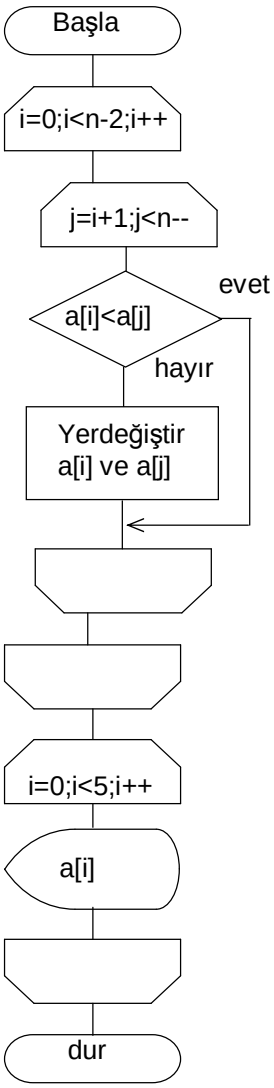
- **Değişken Tablosu**

Değişken	tip	Kullanım
a[10]	int	Değerler
i	int	Kontrol değişkeni
j	int	Kontrol değişkeni

Ø **Ekran Görüntüsü**

```
9
8
8
7
:
```

Ø **Akış Diyagramı**



Ø **Program**

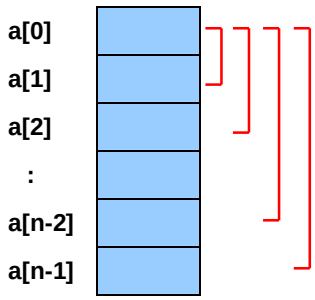
```
/* Örnek 1.3 */

#include <stdio.h>

main()
{
    int  n = 10;
    int  a[10]={3,4,3,2,8,4,6,7,8,9};
    int  i,j,work;

    for (i=0;i<n-1;i++)
    {
        for (j=i+1;j<n;j++)
        {
            if(a[i] < a[j])
            {
                work=a[i];
                a[i]=a[j];
                a[j]=work;
            }
        }
    }

    for(i=0;i<n;i++)
       printf("%d\n",a[i]);
}
```

Ø **Program Açıklaması**

Sıralama aşağıdaki şekilde görüldüğü gibi yer değiştirerek gerçekleştirilir. Sıralama için birçok yol vardır. Biz yandaki metodu kullanacağız.

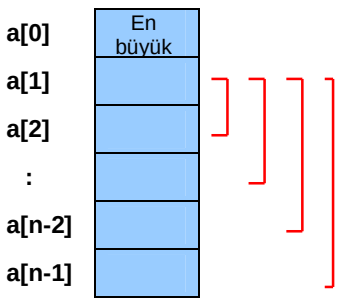
Artan sıralama Azalan sıralama

Ø **1. Adım (en büyük değerin a[0]' a yerleştirilmesi.)**



İlk olarak, a[0] ve a[1] karşılaştırılır. a[1]' in büyük olduğu durumda ikisinin değeri yer değiştirilir. İkinci olarak a[0] ve a[2] yi karşılaştırılır. a[2] > a[0] ise onların değeri yer değiştirilir. Karşılaştırmalar a[0] dan a[n-1] değeri olan son elemana kadar devam edecektir. Son olarak en büyük değer a[0] içerisinde yer alacaktır.

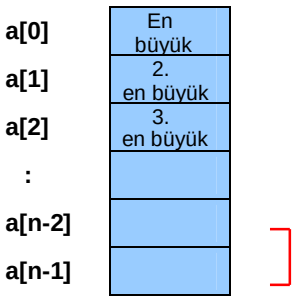
Ø **2.Adım (ikinci en büyük sayının a[1] içine yerleştirilmesi)**



İkinci aşamada, a[1] ve a[0] haricindeki diğer bütün değerlerle karşılaştırmalar yapılır. Bu değerler [2] den a[n-1] e kadar olan değerlerdir. Bu şekilde, a[1] ikinci en büyük sayıya sahip olan alan olabilir.

Üçüncü aşama ve dördüncü aşama aynı durumdadır

Ø Son Adım



Son aşama a[n-2] değeri için gerçekleştirilmelidir. Bu değer ve a[n-1] değeri için karşılaştırma ve yer değiştirme işlemleri anlaşılabilir.

Yukarıdaki basamaklar dış döngü (ilk döngü) yardımıyla gerçekleştirilir. Bu döngüde, kontrol değişkeni i, 0 dan (a[0]' ın ilk adımı için) n-2 (a[n-2] için son basamak) ye kadar değiştirilebilmelidir. Her adımdaki karşılaştırmalar 2. döngü ve iç döngü ile gerçekleştirilir. Kontrol değişkeni j burada, i+1 den n-1 e kadar değiştirilebilmelidir. Tüm bu durumları bütün basamaklar için kontrol etmelisiniz.

a[i] ve a[j] arasındaki değerlerin yerlerini şu şekilde değiştiremeyiz

```
a[i]=a[j];
a[j]=a[i];
```

Bu şekilde, a[i] değeri kaybolacaktır. a[i]' nin değerini geçici olarak saklayabilmek için şu ifadeleri kullanmalıyız.

```
  k = a[i];
a[i] = a[j];
a[j] = k;
```

1.2. İki Boyutlu Dizi

Örnek 1.4.

Sınav sonuçlarını iki boyutlu bir diziden okuyan programı yazınız.

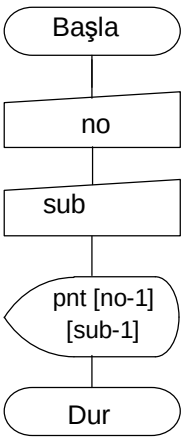
Ø **Ekran Görüntüsü**

```
Öğrenci numarası ==>2
Hangi Dersin Notu ?
1--Matematik   2--Tarih   3--İngilizce
  ====>2
Notu  95.
```

Ø **Değişken Tablosu**

Değişken	Tür	Kullanım
pnt[30][3]	int	Sınav Notu pnt[no][sub]
no	int	Öğrenci numarası 1—Matematik 2—Tarih 3—İngilizce
sub	int	Ders numarası

Ø **Akış diyagramı**



NOT :
Bundan sonra, akış diyagramlarındaki bazı ifadeleri ihmal edeceğiz. Bu şekilde program yapısı daha açık hale gelecektir.

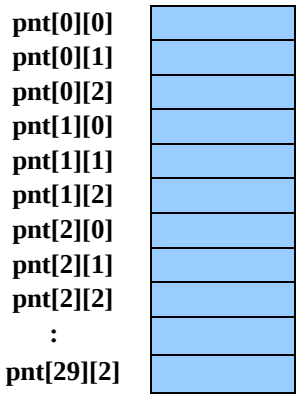
Ø **Program**

```
/* Örnek 1.4 */

#include <stdio.h>

main()
{
    int pnt[30][5]={{80,90,100},                    //--1
                    {80,75,60},
                      :
                    {60,100,65}};
    int    no, sub;

    printf("Ogrenci numarasini giriniz ==>");
    scanf("%d",&no);

    printf("Hangi dersi almak istiyorsunuz? \n";
    printf(" 1—Matematik 2—Tarih      3-Ingilizce\n");
    printf("  ====>");
    scanf("%d",&sub);

    printf("Notu %d. \n", pnt[no-1][sub-1]);        //--2
}
```

Ø Program Açıklaması

İki boyutlu bir dizi şekilde görüldüğü gibi bildirilmiş ve başlangıç değerleri atanmıştır. Ana hafıza içerisinde, dizi elemanları şekildeki gibi yerleştirilmişlerdir.

pnt[alan][alan], ilk alan öğrenci numarası için, ikinci alan ise ders konusu için ayrılmıştır. Alan, 1 ile değil de , 0 dan başladığı için program içerisinde bir değerini çıkarmak durumundayız.



Şekil 1.3: Bellekteki iki boyutlu dizinin elemanları

Şekil 1.3 deki dizi yapısı yerine, şekil 1.4 deki dizi yapısını düşünebiliriz.

	Matematik	**Tarih**	**İngilizce**
Öğrenci 1	pnt[0][0] 80	pnt[0][1] 90	pnt[0][2] 100
Öğrenci 2	pnt[1][0] 80	pnt[1][1] 75	pnt[1][2] 60
:	**:**	**:**	**:**
Öğrenci 30	pnt[29][0] 60	pnt[29][1] 100	pnt[29][2] 65

Şekil 1.4: İki boyutlu bir dizinin yapısı

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Örnek 1.1' deki programı dizi sayısını sabit olmayacak şekilde yazınız. Sayı limitinin üst sınırı 49 olacaktır.
- Aşağıdaki gibi başlatılan bir dizinin yılın başlangıcından itibaren toplam gün sayısını gösteren programı yazınız. Kullanıcı ay ve gün değerlerini girecek ve yıl için döngü kullanabilirsiniz.
- int ay[12]={0,31,59,90,120,151,181,212,243,274,304,334};
- 1' den 50' ye kadar olan sayılardan rasgele olarak kabul eden ve bunları sıralama yöntemi kullanarak azalan şekilde sıralayan programı yazınız.
- Bir dizi içine 10 adet sayı yerleştirdikten sonra bu değerlerin aritmetik ortalamasını bulan programı yazınız.
- Başlatma yapılarak 10 elemanlı bir diziye isteğe bağlı 10 adet integer sayı giriniz. Bu dizi içerisindeki maksimum ve minimum sayıları bulan programı yazınız.

İşlem Basamakları	Öneriler
Ø Değişken tablosunu hazırlayınız. Ø Akış diyagramını çiziniz. Ø Programı yazınız. Ø Yazdığınız programı derleyiniz. Ø Programda hata var ise bunları gideriniz. Ø Ekran görüntüsünü kontrol ediniz.	Ø Programda kullanacağınız değişkenlerin tipini belirleyiniz. Ø Değişken isimlendirme kurallarına dikkat ediniz. Ø Kullanacağınız verinin hangi dizi yapısına uygun olduğuna dikkat ediniz. Ø Dizi sınırlarına dikkat ediniz. Ø Akış diyagramı sembollerinden yararlanınız. Ø Program satırlarının düzenli olmasına özen gösteriniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki ifadeleri doğru [D] veya yanlış [Y] olarak değerlendiriniz.

[ ] 1. **veri [10]** /* dizisi 10 elemanlıdır.*/

[ ] 2. **int a[5]** /* a dizisi bellekte 10 byte yer kaplar.*/

[ ] 3. **no[6]=14** /* no dizisinin 6. elemanı 14 sayısıdır.*/

[ ] 4. **int rakam[100], a , b** /* tanımlaması yanlıştır.*/

[ ] 5. **veri[2] = sayi[2];** /* sayi dizisinin 3. elemanına veri dizisinin 3. elemanı atanıyor.*/

[ ] 6. **for (i = 0; i < 10; i++)** /* ifadesi 10 elemanlı sayı dizisinin tüm elemanlarına sıfır
sayi[i]= 0; /* değeri atar */

[ ] 7.
```
i = 1;
while (i <= 100)
{
    scanf("%d", &fiyat[i]);
    i++;
}                    /*  fiyat dizisinin tüm elemanları için klavyeden değer alınır  */
```

[ ] 8. **int b[5] = {1, 2, 3, 4, 5, 6};** /* ifadesi derlemede hata mesajı verir. ***/**

[ ] 9. **int numara [ ] = {1, 2, 3, 4, 5};** /* ifadesi kullanıldığında küme parantez içindeki */
/* değerleri otomatik olarak kabul eder. */

[ ] 10.
```
gecici= a[i];
a[i] = a[j];
gecici = a[j];  /* ifadesiyle a dizisinin i nolu değeri j nolu değeri ile yer
                   /* değişiyor */
```

DEĞERLENDİRME

Performans testi sonucu "evet", "hayır" cevaplarınızı değerlendiriniz. Eksiklerinizi faaliyete dönerek tekrarlayınız. Tamamı "evet" ise diğer öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Bilgisayar programlamada karakter ve karakter katarlarını doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

Bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapmalısınız.

Ø C programlamada karakter katarları hakkında araştırma yapınız.

2. KARAKTER KATARLARI

Şimdiye kadar genel olarak sayısal veriler ile çalıştık. Karakter ve karakter katarları bilgisayarlarda farklı şekillerde ele alınırlar.Bu öğrenme faaliyetinde karakter tipindeki veriler anlatılacaktır.

2.1. Karakter ve Karakter Kodu

Örnek 2.1.

Klavyeden girilen bir karakteri ve kodunu görüntüleyen programı yazınız

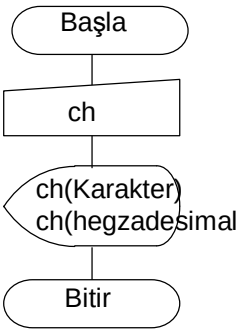
Ø Ekran Görüntüsü

```
Bir karakter giriniz ==>J
Karakter=J   Kod=4A
```

Ø Değişken Tablosu

Değişken	tür	Kullanım
ch	char	Karakterin Saklanması

Ø **Akış Diyagramı**



Ø **Program**

```
/* Örnek 2.1 */

#include <stdio.h>

main()
{
    char ch;

    printf("Bir karakter giriniz ==>")
    scanf("%c",&ch);

    printf("Karakter=%c   Kodu=%X", ch,ch);
}
```

Ø **Program Açıklaması**

Şimdiye kadar öğrendiklerimiz gibi, printf ve scanf fonksiyonlarında tip dönüşüm belirteçlerini kullanacağız. Bir karakter, hafızaya karakter kodu formunda kaydedilir. Bir karakterin kodu %x tip dönüşüm belirteci ile görüntülenebilir.

(1) Karakter Kod tablosu

Karakterleri belirtmek için kullanılacak veriler çok fazladır. (Bunu anlayabilmek için, A karakterini alfabeyi bilmeyen bir yabancıya açıklamaya kalkmanın ne kadar zor olduğunu düşününüz.)Bu yüzden bilgisayarlar her birisi kendi koduna sahip tüm karakterlerin şekillerini içerirler. Sadece 1 byte bir karakteri saklayabilmek için yeterlidir. 1 byte bize 2^8=256 adet farklı karakter üretebilir.

Karakterler için, çeşitli kodlar vardır. Şekil 2.1 yaygın olarak kullanılan bir tablo olan ASCII (American Standard Code for Information Interchange) kod tablosunu göstermektedir. Tablo satır ve sütunlara sahiptir. Karakterlerin sayısal değerinin bulunması için önce sütun daha sonra satır bilgisi okunur. ‘j’ karakterinin kodu şekilde görüldüğü gibi onaltılık sistemde olarak 4A değerine eşittir.

Sütun 0 ve 1 in kodu karakter kodları için değildir. Bunlar ekran kontrolü ve veri transfer kontrolü için kullanılır. Bütün bunlar kontrol kodu olarak ifade edilirler. ASCII tabloda 3. sütunda gösterilen sayısal değerler aynı zamanda karakter koduna sahiptirler. Örneğin ‘3’ karakterinin kodu 33 sayısıdır.

ü / ı								
0	NUL	DLE	Space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	TAB	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|

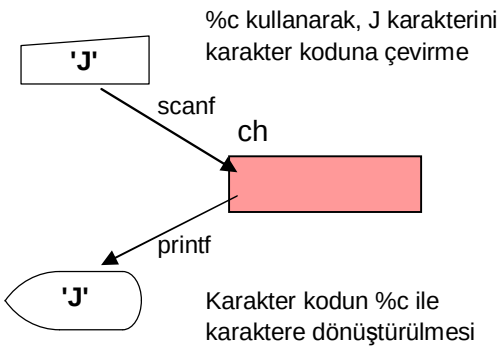
Gölgelendirilmiş bölümler daha sık kullanılır. Şekildeki örneğe gore ‘J’ nin karakter kodu $4A_{16}$ değeridir.

Şekil 2.1: ASCII karakter kodları

Karakterden koda dönüştürmede veya tersi işleminde scanf ve printf fonksiyonlarında kullanılan %c tip dönüşüm belirteçleri kullanılmıştır.

(2) Karakterlerin printf ve scanf fonksiyonunda kullanımları

```
char ch;
scanf("%c",&ch);    //--1
printf("%c",ch);    //--2
```



Şekil 2.2: Karakter alma ve gösterme

1. gösteriminde, yazılan karakter %c kullanılarak karakter koda dönüştürülmüştür. Daha sonra karakter printf fonksiyonunda %c komutuyla gösterilmektedir.

%c yerine %X kullanarak karakter kodu hegzadesimal formda gösterebiliriz.

printf("%X",ch); veya %d ile, desimal olarak printf("%d",ch);

(3) Karakter sabiti

Bir karakteri tek tırnak arasına aldığımızda, bu karakter bir karakter sabiti olarak kabul edilirler. Karakter sabiti karakter koduna sahiptir. Örneğin;

ch = ’D’; komutu ch değişkenine D nin kodunu aktarır.

Karakter katarları aşağıdaki şekilde başlatılabilir.

char char[ ]={'A','B','C'};

Tek tırnak işareti, karakterin, karakter kodundan önde geldiği durumlarda da kullanılır.

Bir problemde scanf fonksiyonunda %c kullanıldığında, bazen beklenilmeyen sonuçlar doğabilir.
Örneğin, Aşağıdaki gibi 2 tane scanf fonksiyonu yazdığımızda,

```
scanf("%d",&fiyat);
scanf("%c",&kayit);
```

LF kodu (Kontrol kodu- 0A ASCII), desimal sayı girildikten ve enter tuşuna basıldıktan sonra üretilen kod ch değişkenine atanır. Bu yüzden, sonuç olarak ikinci scanf fonksiyonu atlanır.

Bu durumu engellemek için ikinci scanf fonksiyonunda şöyle bir düzenleme yapmalıyız.

```
scanf("%c%c",&tutar,&kayit);
```

Bu durumda LF kodu dummy. değişkenine atanır.Eğer karışık gelmez ise bu kodda kullanılabilir.

```
scanf("%c%c",&kayit,&kayit);
```

2.2. Karakter Katarları

Bir karakter dizisinde saklanan karakter katarına string adı verilir.

Örnek 2.2.

10 karaktere kadar bir string ifadeyi klavyeden okuyup, her bir karakteri ve kodunu görüntüleyen programı yazınız.

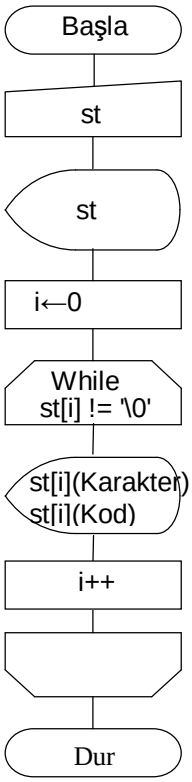
- **Ekran Görüntüsü**

```
String ifade giriniz ==>bitir
String=bitir
Karakter=s Kod=73
Karakter=t Kod=74
Karakter=o Kod=6F
Karakter=p Kod=70
```

- **Değişken tablosu**

Değişken	Tür	Kullanım
st[11]	int	String saklama
i	int	Kontrol değişkeni

Ø Akış Diyagramı



Ø Program

```
/* Örnek 2.2 */

#include <stdio.h>

main()
{
    char st[11];
    int  i;

     printf"string ifade giriniz==>");
    scanf("%s",st);

    printf("String=%s\n",st);
    i=0;
     while(st[i] !=' \0')
    {
        printf("Karakter=%c Kod=%X \n",st[i]);
        i++;
     }
}
```

Ø Program Açıklaması

String ifade printf ve scanf fonksiyonlarında %s ile kullanılır. Bir karakter katarında tutulan string ifadenin elemanlarına ulaşabiliriz. Detaylı açıklamalar program yapısında anlatılacaktır.

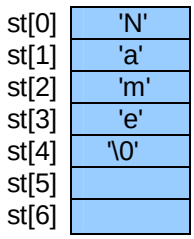
(1) Bir karakter dizisindeki string ifade

String ifadedeki her bir karakter, 1 karakterlik boşluğa gerek duyar. Bu yüzden string ifade için karakter katarını şu şekilde hazırlarız.

char st [7];

C dilinde, String ifadenin sonu 0 ile sonlandırılır. 0 değeri ASCII kod tablosunda BOŞ(NULL) değeridir. Bu karakter, NULL karakter olarak adlandırılır ve '\0' ile gösterilir. Hafızanın yapısının gerektirdiği '\0' değer için, karakter dizisinin boyutu (string ifadenin boyu + 1) şeklinde ayarlanmalıdır. Bir string ifade için kullanılan dizide, \0 değerinden sonraki değerlerin string ifade için bir önemi yoktur.

Şekil 2.3 de bu durum st[5] ve st[6] alanlarıdır.



Şekil 2.3: Karakter katarında string ifade

(2) printf fonksiyonu ile string ifadenin yazdırılması

Bir karakter dizisindeki string ifadeyi şu şekilde gösterebiliriz.

printf(“%s”,st);

Tip dönüşüm belirteci olarak %s ve dizi ismi olarak st kullanılmıştır. Dizi ismi, parantez ve parametresiz kullanıldığında, dizinin en üst adresine sahip olur. Böylce %s tip dönüşüm belirteci ile printf fonksiyonu, birer birer en üstten başlamak üzere \0 kadar olan bütün değerleri yazar.

(3) scanf fonksiyonu ile string ifadeler.

scanf fonksiyonundaki tip dönüşüm belirteci printf fonksiyonundaki ile aynıdır. Bu yüzden şu şekilde yazılabilir.

scanf(“%s”,st);

st dizi isminin kullanılmasından dolayı , & adres operatörüyle aşağıdaki şekilde yazamazsınız;

scanf("%s",&st);

Takip eden kod hatalı değildir fakat biraz hantal bir yapıya sahiptir.

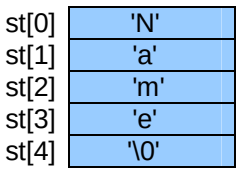
scanf("%s",&st[0]);

scanf fonksiyonu, dizinin en üst noktasından itibaren değerleri okur ve sonuna \0 değerini ekler. Bir string ifadeyi içerisindeki boşluklarla girdiğinizde, scanf fonksiyonu sadece birinci boşluğa kadar okuyabilir.

(4) String sabiti ile başlatma

Eleman değerini yazmak için, karakter dizisinin başlatılması şu şekilde de gerçekleştirilebilir.

char st[ ]="Name";



Şekil 2.4: String ifadesiyle başlatma

String ifadede başlatma işlemini şu şekilde bir tanımlama ile yapamayız.

char st[5];
st="name";.

Bir string ifadenin başka bir diziye kopyalanması yada karakter dizisine string sabiti olarak ayarlanması için **strcpy** fonksiyonu kullanılır. Bu komutu saha sonra çalışacağız.

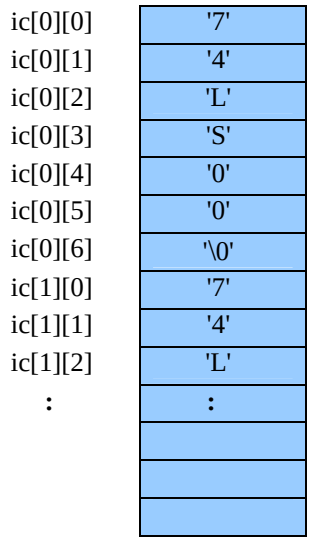
Karakter katarı str yi şu şekilde başlatabiliriz,

char str[]=” artık kendi programlarımı yazabiliyorum.”

(5) İki boyutlu diziler ve string ifadeler

İki boyutlu karakter dizisi bir string’ten daha fazlasını saklayabilir. İlk parametre, 0 dan itibaren string ifadenin sırasını gösterir. İkinci parametre 0 dan itibaren string ifadedeki karakterlerin sırasını göstermektedir. Her string ifade, sonunda **\0** değerine sahip olmalıdır.

Şekil 2.5 de, "74LS00", ve "74LS04" entegrelerin isimleri bellekte saklanmaktadır.



Şekil 2.5: Bir string ifadenin başlatılması

İki boyutlu karakter katarı aşağıdaki şekilde ifade edilebilir.

char dizi_isim[string sayısı][Stringlerin maksimum uzunluğu+1];

String ifadeye ulaşmak için, her string ifadenin en üst adresini kullanırız. ”74LS00” değerini şekildeki gibi göstermek için örneğin, printf fonksiyonu aşağıdaki gibi kullanılır.

printf("%s",&ic[0][0]), bir başka ifadeyle,

printf("%s",ic[0]); (& ve ikinci parametreyi kullanmadan.)

Tek boyutlu bir dizideki dizi ismi olarak en üst adresi gösterebiliriz.

ic[0], ic[0][0]-ic[0][6] nin en üst adresini ifade etmektedir. Benzer şekilde, scanf fonksiyonu aşağıdaki gibi yazılabilir.

scanf("%s",ic[0])

Aşağıdaki program, iki boyutlu bir karakter dizisinin nasıl başlatıldığını gösterir.

Ø **Program**

```
/*  İki boyutlu bir dizi için program örneği*/

include <stdio.h>
main()
{
     int i;
     char ic[3][7]={"74LS00","74LS02","74LS04"};

     printf("Hangi entegreyi siparis etmek istiyorsunuz? \n");
     printf(" 1.%10s 2.%10s 3.%10s\n",ic[0],ic[1],ic[2]);
     printf (" numarasını giriniz (1-3)   ==>");
     scanf ("%d",&i);

     printf ("siparis etmis olduğunuz entegre.. %s. \n", ic[i-1]);
}
```

Ø **2.2.1. String Fonksiyonları**

Dizi içerisindeki çalışma yapabilmek için gerekli bazı komutlar aşağıdaki şekil 2.6 de verilmiştir. Bu fonksiyonları kullanmak için programımıza **string.h** kütüphanesini ilave etmeliyiz.

#include<string.h>

Fonksiyon	Amaç	Örnek
strcpy(a,b)	A dizisindeki bir stringi b dizisine kopyalar	**strcpy(a,b)**
strlen(a)	a dizisindeki string ifade için String uzunluğunu bulur(karakter numarası)	**n=strlen(a)**
strcat(a,b)	b dizisindeki bir string'i a dizisine ekler	**strcat(a,b)**
strcmp(a,b)	a dizisindeki string ifadeyle b dizisindeki string ifade karşılaştırılır.Dönen değer şu şekildedir. a > b → pozitif değer a = b → 0	**char a[ ]="aaa";** **char b[ ]="bbb"** **if(strcmp(a,b)<0)**

Şekil 2.6: String fonksiyonları

Örnek 2.3

Bir dizideki string ifadeyi başka bir diziye kopyalayan programı yazınız. İlk dizinin string ifadesi klavyeden girilecektir.

Ø Program

```
/*  Örnek 2.3  */

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

main()
{
        char a[11],b[11];
        int  i;

        printf("Bir string giriniz => ");
        scanf("%s",a);

        strcpy(b,a);
        printf("  a[ ]=%s\n",a);
        printf("  b[ ]=%s\n",b);
 }
```

Ø Ekran Görüntüsü

```
Bir string giriniz => Başla
    a[ ]=başla
    b[ ]=başla
```

Program Açıklaması

String fonksiyonu olan strcpy() string ifadeyi kopyalamak için kullanılır. Programda "a" string ifadesi "b"ye kopyalanmıştır.

Örnek 2.4.

10 karakterlik iki kelimeyi kabul edip, hangisinin sözlükte öncelikli yeraldığını bulan programı yazınız.

Ø Ekran Görüntüsü

```
1. kelime     => kitap
2. kelime     => elma
elma kelimesi sözlükte daha önce gelir
```

Ø Program Açıklaması

Burada, if komutu için, strcmp( ) fonksiyonundan dönen değer kullanılmıştır. Dizinin ilk eleman eğer sözlükte daha önce geliyorsa strcmp( ) fonksiyonu negatif bir değer döndürür. Bu string fonksiyonunda, string sabiti şu şekilde de kullanılabilir.

strcpy(a,"cem"); : "cem" karakter gurubunu a dizisine aktarır.

strcmp(a,"ccc"); : a dizisinin içindeki karakter gurubu ile "ccc" karakterlerini kararlaştırır.

Ø **Program**

```
/*  örnek 2.4  */

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

main()
{
        char wd1[11],wd2[11],wdf[11];
        int  i;

        printf("1. kelime => ");
        scanf("%s",wd1);

        printf("2. kelime =>");
        scanf("%s",wd2);

        if(strcmp(wd1,wd2)<0)
              strcpy(wdf,wd1);
       else
      strcpy(wdf,wd2);

       printf("\n%s kelimesi sozlukte daha once gelir\n",wdf);
}
```

Bir string sabitini çift tırnak arasında yazarsak , bu string' in hafızadaki en üst değerine sahip olur. Dizi ismi karakter dizisinin en üst adresine sahip olmasından dolayı, string fonksiyonlarının bu argümanları için değer aldığını söyleyebiliriz. Bu nedenle, string fonksiyonu, bu adresteki karakterden başlar \0 değerini görünceye kadar işlemeye devam eder.

Aşağıda görülen kodu deneyiniz.

printf("Karakter gurubunun adresi=%X","boy");

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Örnek 2.1' i düzenleyerek, girilen 'e' değerini görünceye kadar tüm süreci tekrar ediniz. Programdaki bazı karakterler için karakter kodunu kontrol ediniz.
- Bir cümlede kullanılan karakterlerin sayısını bulan programı yazınız. Karakterler klavyeden girilecek ve 0 girilene kadar değiştirilebilecek şekilde düşünülecek
- İki string ifadeyi kabul edip daha sonra bunları birleştiren ve birleşmiş halini ve toplam karakter sayısını görüntüleyen programı yazınız.
- 50 adete kadar girilen rastgele kelimeleri alfabetik sıraya sokan programı yazınız.
- Aşağıdaki bir string ifadedeki kelimeleri iki boyutlu bir diziye teker teker aktarıp daha sonra bunları alfabetik sırada görüntüleyen programı yazınız.

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

main()
{
   char str = "Calısma seklimizde önemli degisiklikler olacak  ";
   char word[50][15];
         :
```

İşlem Basamakları	Öneriler
Ø Değişken tablosunu hazırlayınız. Ø Akış diyagramını çiziniz. Ø Programı yazınız. Ø Yazdığınız programı derleyiniz. Ø Programda hata var ise bunları gideriniz. Ø Ekran görüntüsünü kontrol ediniz.	Ø Programda kullanacağınız değişkenlerin tipini belirleyiniz. Ø Değişken isimlendirme kurallarına dikkat ediniz Ø Karakter katarı sınırlarına dikkat ediniz. Ø Akış diyagramı sembollerinden yararlanınız. Ø Program satırlarının düzenli olmasına özen gösteriniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki ifadeleri doğru [D] veya yanlış [Y] olarak değerlendiriniz.

[ ] 1. **$5E_{16}$** karakter kodunun karşılığı " ^ " sembolüdür.

[ ] 2. "**a"** harfinin karakter kodu **41_{16}** sayısıdır.

[ ] 3. **printf("%X",ch);** /* komutu karakter kodunu gösterir. */

[ ] 4. **char [ ]={ A,B,C };** /* ifadesiyle karakter katarları başlatılır. */

[ ] 5. **char isim[ ] = "Ali";** /* 3 elemanlı bir string ifadesidir. */

[ ] 6. **scanf("%s",&kelime);** /* komutu ile klavyeden karakter katarı alınır. */

[ ] 7. **strcat(a,b)** /* b dizisindeki string ifadeyi a dizisine ekler. */

[ ] 8. **n=strlen(a)** /* a dizisindeki karakter sayısını n değişkenine atar. */

[ ] 9. **strcmp(a,b)** /* fonksiyonunda a<b ise a kelimesi sıralamada b den önce gelir.*/

[ ] 10. **printf("%s",kelime[3]);** /* iki boyutlu karakter katarındaki 4. string ifadeyi tanımlar */

DEĞERLENDİRME

Performans testi sonucu "evet", "hayır" cevaplarınızı değerlendiriniz. Eksiklerinizi faaliyete dönerek tekrarlayınız. Tamamı "evet" ise diğer öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–3

AMAÇ

Bilgisayar programlamada disk dosyalarının yazma, okuma ve ekleme işlemlerini doğru bir şekilde yapabileceksiniz.

ARAŞTIRMA

Bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapmalısınız.

- C programlamada dosyalar hakkında araştırma yapınız.

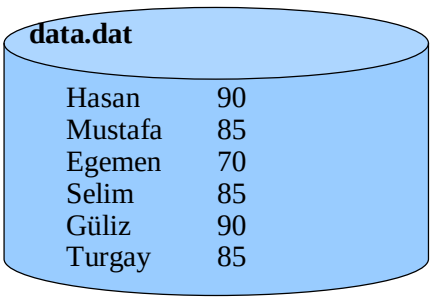
3. DOSYALAR

Dosyalar programların dışında kalan verileri taşırlar. Dolayısı ile dosyalar ile verilerde yapılan değişikliklere bağlı olarak programları değiştirmeye ihtiyaç duymayız veya programı çalıştırmak için her zaman klavyeden veri girişi yapmaya gerek yoktur. Dosyaların kullanımına ait yazım kurallarına ait detayları anlamak bu seviyede oldukça zor olabilir. Dolayısıyla sadece dosyaların nasıl kullanılacağına yoğunlaşmalıyız.

3.1 Dosyanın Okunması

Örnek 3.1

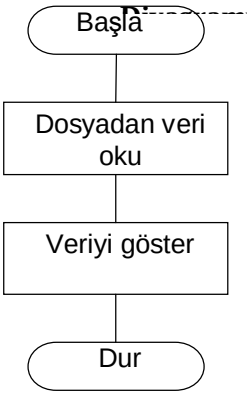
Sağda görülen dosyayı okuyan ve okuma işleminden sonra görüntüleyen programı yazınız



- **Değişken Tablosu**

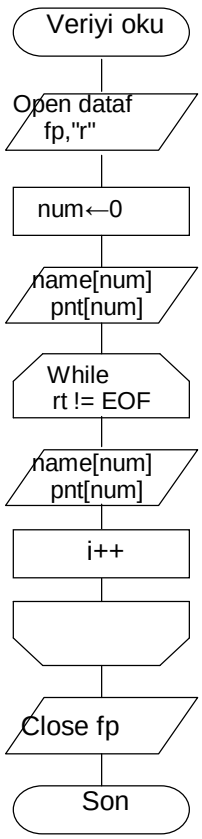
Değişken	tip	Kullanımı
fp	FILE *	Dosya işaretçisi (pointer)
name[100][11]	int	Öğrenci isimleri
pnt[100]	int	Sınav notları
num	int	Öğrenci numarası
rt	int	Dönen değer
i	int	Kontrol değişkeni

Ø Genel Akış Diyagramı

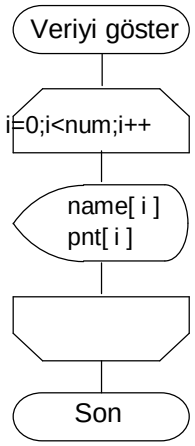


Bu bölümde ilk olarak genel akış diyagramlarını kullanacağız ve daha sonra bu diyagramların mantığının yeni ve daha karışık olmalarından dolayı bazı detaylarını çizeceğiz. Genel akış diyagramına verilen örnekte görüldüğü gibi, programın tümüne ait bir görümüne sahiptirler.
Akış diyagramlarının ayrıntısı, genel akış diyagramında görülen her bir kutunun için gösterilir. Bu yolla, geniş programların yapısını gösterebiliriz ve bunları gruplayarak iyi programlar yazabiliriz

(1) Dosyadan veri okuma



(2) Veriyi gösterme



Ø Program

```
/*  örnek 3.1  */

#include <stdio.h>

void main()
{
   FILE *fp;
   char name[100][11];
   int  pnt[100], num, rt, i;

   fp=fopen("dataf.dat","r");
   num=0;
   rt=fscanf(fp,"%s%d",name[num],&pnt[num]);
   while(rt != EOF)
   {
        num++;
        rt=fscanf(fp,"%s%d",name[num],&pnt[num]);
   }

   for(i=0;i<num;i++)
     printf("%-10s  %3d\n",name[i],pnt[i]);

   fclose(fp);
}
```

Ø Program Açıklaması

(1) Veri dosyasını oluşturma

Örnekte gösterilen veri dosyası, giriş yapan programlar için kullanılan aynı yazım editörleri ile oluşturulabilir. Yazım editörleri erişilebilen bu dosyalar metin dosyalar olarak ve veri taşıyan bu dosya, metin veri olarak isimlendirilir.Burada sadece metin veri dosyalarını çalışacağız. Bununla birlikte C dili ile ikili sistemdeki veri dosyalarına erişebiliriz.

(2) Dosya açma, kapatma ve işaretçi (dosya işaretçisi)

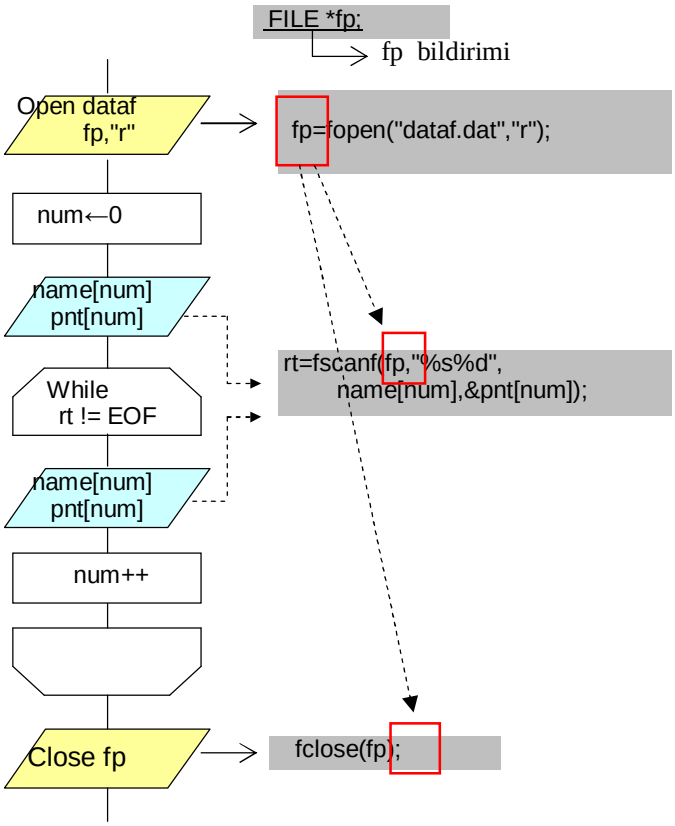
Bir programda dosya kullanmak için, hazırlık amacıyla açmak zorundayız ve kullanımdan sonra kapatmalıyız. "Açmak" kelimesinin anlamı sistemin ele alınan dosyada gerekli hazırlığı yapmasıdır. C dilinde fopen fonksiyonu bu işlemi yapar. fopen fonksiyonu iki argümana sahiptir.

Bunlardan biri dosya ismidir, diğeri ise şekil 3.2.'de gösterilen mod'dur. fopen fonksiyonu dosyaya erişmek için ihtiyaç olan erişim parametreleri bilgisini geri döndürür.

Bu bilgi dosya işaretçisi (file pointer) (FILE * şeklinde tanımlanmış) olarak isimlendirilen bir değişkene atanmak zorundadır. Dosya işaretçisi (file pointer) üzerinde

detaylı şekilde çalışmayacağız.Veri dosyasına dosya ismi ile değil, dosya işaretçisi (bu programda fp) ile erişilir.

Dosya, dosya işaretçisinin argümanı olan fclose fonksiyonu kullanılarak kapatılır. Bu işlemden sonra dosya işaretçisi bir diğer dosya için tekrar kullanılabilir.



Şekil 3.1: Dosyadan veri okuma

Mod	Kullanımı
"r"	Veri okuma
"w"	Veri yazma(dosya veriye sahipse üzerine yazma)
"a"	Veri ekleme (dosya yok ise yeni bir dosya oluşturma)

Şekil 3.2: Fopen fonksiyon modları

(3) fscanf fonksiyonu ve dosyadaki veri sonunun kontrolü

fscanf fonksiyonu, aşağıda tekrar gösterilen programda bir dosyadan satır halinde okuma yapmak için kullanılır.

```
rt   =   fscanf(fp,"%s%d", name[num],&pnt[num]);
```

"fscanf" fonksiyonunun "scanf" fonksiyonundan tek farkı, fscanf argümanı olan bir dosya işaretçisine sahip olmasıdır ve bu işaretçi stdio.h dosyasında EOF (dosyanın sonu) olarak tanımlanmış değeri geri döndürür. Integer tipinde bir değişkene (bu programda rt), dönen değerin atanması suretiyle, tüm şartlarda dosyadaki kullanılmış olan veriyi kontrol ederiz.

```
rt=fscanf(fp,"%s%d",name[num],&pnt[num]);
while(rt != EOF)
{
   num++;
   rt=fscanf(fp,"%s%d",name[num],&pnt[num]);
}
```

3.2 Dosya Yazma

Örnek 3.2

Klavyeden isimlerin girildiği ve bir dosyada saklandığı bir program yazınız.

Ø Ekran

```
Isim giriniz (Son için e) ==> Ömer
Isim giriniz (Son için e) ==> Emine
      :
Isim giriniz (Son için e) ==> e
```

Ø name

Ø Genel Akış

Ø Değişken

Değişken	tip	Kullanım
fp	FILE *	file pointer
name[100][11]	int	İsimler
num	int	Sayı
rt	int	Dönen değer
i	int	Kontrol değişkeni

Ø **Program**

```
/*  örnek 3.2  */

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

void main()
{
   FILE *fp;
   char name[100][21];
   int  num, i;

   num=0;
   printf("Isim giriniz (Son için e) ==>");
   scanf("%s",name[num]);
   while( strcmp(name[num],"e") != 0)
   {
      num++;
      printf("Isim giriniz (Son için e) ==>");
      scanf("%s",name[num]);
   }

   fp=fopen("namef.dat","w");

   for(i=0;i<num;i++)
      fprintf(fp,"%s\n",name[i]);

   fclose(fp);
}
```

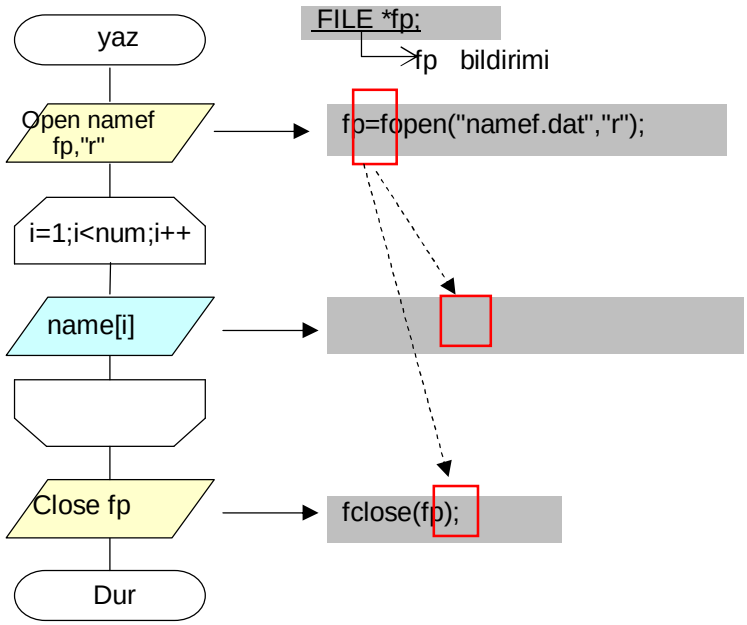
Ø **Program Açıklaması**

Bu programda yeni olan, sadece dosyaya yazım yapılan bölümdür. Dosyanın okunabilmesi için, dosya fopen fonksiyonu ile açılmak zorundadır. Bu anda mod, yazma anlamına gelen "w" 'dır. Bu fp dosya işaretçisi kullandıktan sonra, dosyayı düzenleyebiliriz.

Dosyaya yazmak için fprintf fonksiyonunu şu şekilde kullanırız.

fpintf(**fp**,"%s%d", name[num],&pnt[num]);

Bu fonksiyonun printf'den farkı sadece ilk argümanın dosya işaretçisi olmasıdır.

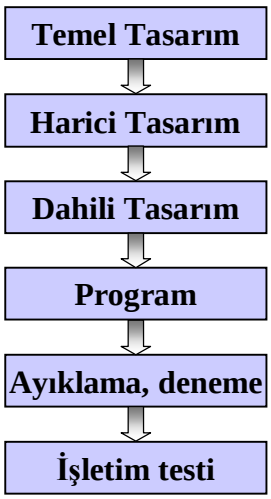


Şekil 3.3: Dosyaya veri yazdırma

3.3 Sistem Gelişimi ve Bir Veriye Erişim

Programlama çalışırken en önemi tecrübe, kendi düşüncelerinizle doğan bir program oluşturmaktır. Sadece bu deneyim ile programlamayı en iyi şekilde öğrenebilirsiniz. Bu bölümde basit veriye erişim sistemleri ile uygulamalar yapacağız. Daha önce sistem gelişiminin nasıl yapılması gerektiğini çalışmak zorundayız.

3.3.1. Sistem Gelişim Süreci



Şekil 3.5:Sistem gelişim süreci

Büyük sistemlerin, uzun süreli gelişim süreci şimdiye kadar uyguladığımız küçük programların yazım aşamalarına benzer.

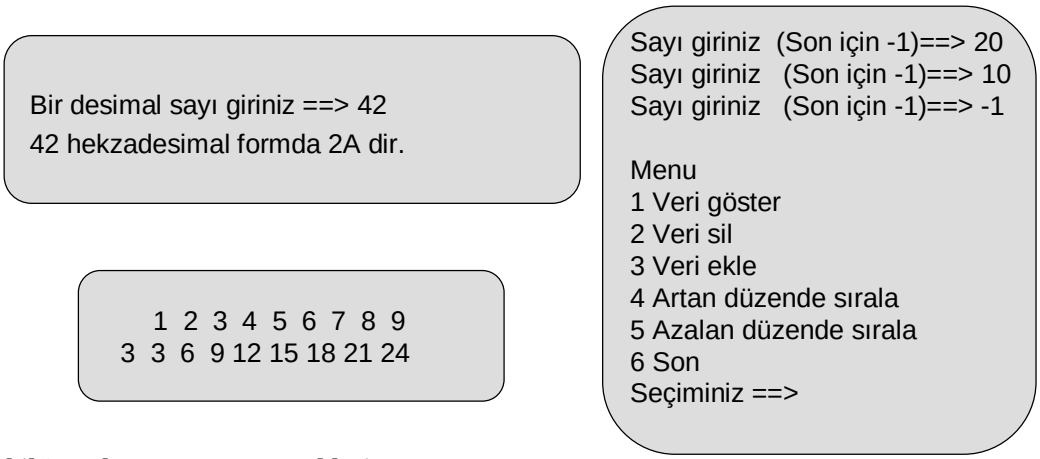
(1) Temel Tasarım

İlk olarak ne yapmak istediğimize ve sistemin içeriğine karar vermek durumundayız. Bu, nihayetinde büyük bir sisteme veya göreceli şekilde küçük olabilecek daha sınırlı sistem içeriğine dönüşecek, karmaşık sistemler olabilir. Kaba özellik burada listelenmelidir. Bunun için sistemin kapsayacağı içerik ortaya konur.

Bir firmada proje çalışması yapılırken bu bölüm, araştırmalar ve dikkatli kontrollerle oluşturulmalıdır.Çünkü bu sistemin gelişim planının maliyetinin ne kadar olacağını ve ne kadar devam edeceğini verir.

Ø (2) Harici Tasarım

Harici tasarım, aynı zamanda giriş ve çıkış tasarımı şeklinde de isimlendirilir. Sistemde kullanıcının ne tür girişler yapacağını ve ne gibi çıkışlar elde edeceğini belirlemek zorundasınız. Ekran tasarımı, temel tasarımda oluşturulan özelliklere göre burada şekillendirilmelidir. Bu bölüm, sadece genel tanımlamalara sahip olan özelliklerden daha fazla sisteminizi ortaya çıkarır çünkü birçok şey ekran tasarımında belirleyici olacaktır.



Şekil 3.5:Ekran tasarımı örnekleri

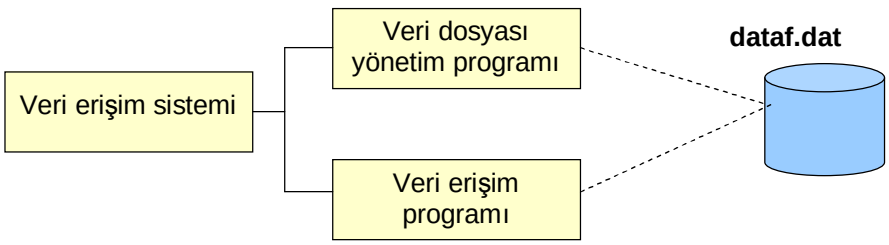
Özellikle harici tasarım, uygulamalarımızda çok önemlidir. Çünkü burada yapılacak her hangi bir değişiklik bütün birimleri etkiler. Harici tasarımdaki bazı değişiklikler, dosya yapısını veya program sayısını değiştirebilir ve bunları oluşturan yapı sistem için gereklidir.

Harici tasarımda programın gerçekleşmesinin nasıl olacağını göz önünde tutulmamalıdır. Bu dahili tasarım sürecinde yapılmalıdır. Öncelik, sistemde için arzu edilen giriş ve çıkışların ne biçimde olacağına verilmelidir.

Ø (3) Dahili Tasarım

Dahili tasarımda, sistemimizde kaç tane program olacağı ve kullanacağımız dosya türleri belirlenir.

Şekil 3.6 basit veri erişim sistemini göstermektedir.



Şekil 3.6:Dahili tasarım

Bu şekil sistemin iki tane programa sahip olduğunu göstermektedir. Biri, dosyaya veri girmek ve dosyanın içeriğini düzenlemek içindir. Diğeri ise veriye erişim içindir. Kullanılan dosyaya her iki programla da erişilebilir. Dosya yapısı aşağıda gösterilen şekilde belirtildiği gibi olmalıdır.

[sinav.dat]

Nesne	Tipi	Kullanımı
name	string	Öğrenci adı
point	int	Sınav notları

Şekil 3.7:Dosya yapısı

Şekil 3.7 'de ikinci sütundaki kutuların, farklı programlarda bu şekilde olması gerekmez. Bunlar çoğu durumda bir program olabilir. Menü bölümü seçim için kullanılır. Fakat fonksiyonları kesinlikle ayrılmış olmak zorundadır ve diğer bölümleri etkilememelidir.

- **(4) Program Tasarımı ve Oluşumu**

Bu bölümde ne yapmamız gerektiği, önceki süreçte zaten oluştu. Değişken tablosu ile ihtiyaç olan değişkenleri listelemeli ve genel akış diyagramı ve ihtiyaç halinde ayrıntılı akış diyagramı ile temel yapıyı belirlemelisiniz.

- **(5) Ayıklama ve Deneme, İşletim Testi**

Program oluşumundan sonra bazı verilerle denenmelidir. Uygulamalı sistemlerde, uygulamacının yaptığı deneme sonrasında gerçek kullanımından önce, işletim testi olarak isimlendirilen ve gerçeğe uyan durumlarla test edilmelidir.

3.3.2 Veri Erişim Sistemine Örnek

Burada tüm sistem gelişim sürecini kullanarak, basit hastane arama sistemini çalışacağız. Sonrasında bunun temelleri üzerine kendinize ait küçük projelerle geliştireceksiniz.

- **(1) Temel tasarım**

Bazı düşünce veya fikir alışverişleri ile sistemin nasıl olacağı belirlenir. Sistem içeriği ve detayları ortaya konulmalıdır. Bu işlemlerden sonra işimiz çok daha kolaylaşacaktır. Özellikler aşağıda gösterildiği gibi yazılmalıdır.

Özellikler

1.Kullanıcı şehir ve ilçe bölümleri ile hastaneleri araştırır
2.Sistem Konya ve İzmir'deki hastaneleri kapsamaktadır
3.Kullanıcı hastane bölümlerini aşağıdaki 4 departman üzerine araştırabilir.
    1) Dahiliye Bölümü 2) Travma cerrahisi
    3) Pediatri Bölümü 4) Doğum kliniği
4.Kullanıcı program sona ermeden önce devamlı şekilde araştırma yapabilir.

Uygulamamız için sistemin içeriği makul olmalıdır. Aksi takdirde çok büyük ve çok karmaşık bir hal alacaktır.

Ø (2) Harici Tasarım

Ekran tasarımı yukarıdaki özelliklere göre yapılır. Oluşturma sürecinde çok daha detaylı olarak düşünmek zorundasınız. Arzu edilen kullanıcı işlemleri için nasıl bir program oluşturacağınıza yoğunlaşmalısınız. Aşağıdaki şekil hastane araştırma sistemimizin ekran tasarımını göstermektedir.

```
 Şehir seçiniz.
 1.  İzmir  2. Konya
   ==> 1

İlce seciniz
  1. Alsancak   2. Konak  3.Bornova
    ==> 3

Departman seçiniz.
  1. Dahiliye Bölümü
  2. Travma Cerrahisi
  3. Pediatri Bölümü
  4. Doğum Kliniği
   ==> 1

  Alsancak'ta Dahiliye Bölümüne sahip hastaneler aşağıda
listelenmiştir.
    Adı              Adres
    ABC    Hastane    1379Sokak
    ZZZ    Klinik      1234Sokak
                :
Tekrar arama( 1--evet/ 2--hayır) ==> 1
Şehir seçiniz .
1.  İzmir  2. Konya
    :
```

Ø (3) Dahili Tasarım

Dahili tasarımda programcıların yetenek ve tecrübeleri, tüm resmin başarılı bir şekilde çizilmesinin nasıl başarılacağında en önemli faktördür.

a) Program Verileri

Verinin depolandığı alanı (programda veya dosyada) ve veriye nasıl erişilmesi gerektiğini belirlemek zorundasınız. Verilerin ayırt edilmesi için kodlar çok önemli rol oynar. Bu kısımda kodların detayları hakkında söz etmediğimiz halde, iyi kodlama programı basitleştireceği için, dikkatli davranmalısınız.

Aşağıdaki bölümde hastane sistemimiz ele alınmıştır.

Şehir **è** Sistem sadece iki şehir ele alınacağı için, programdaki veriler bu şehirlerle ilgili olmalıdır.

Departmanlar **è** Sistemde 4 departmandan bahsedildiğine göre departman isimlerini programda dizi kullanarak saklayacağız. Daha sonra departmanlara bu yöntemle erişebiliriz

İlçe **è** İlçelerin sayısı fazladır ve bazen artabilir. Dolayısı ile dosyada saklarız.

Basit bir şekilde 1,2,3 gibi kodlarla ve ilçe isimleriyle izmtown.dat ve kontown.dat şeklinde iki dosya oluştururuz.

İzmtown.dat

```
1 Alsancak
2 Bornova
3 Konak
  :
```

Alanların içeriği

no	Nesne	Tipi
1	Town code	İnt
2	Town name	String

Hastane verileri **è** Hastane sayısı bir hayli olacağı ve kolayca değişebileceği için her bir şehir için oluşturulan dosyalarda saklayacağız. Belki de bu sistemin en zor bölümü, hastanelerin sahip olduğu bölümlerin gösterilmesi için verilerin nasıl hazırlanması gerektiğidir. Bunun için birden fazla ihtimal düşünebilirsiniz. Aşağıda görüldüğü gibi en basit olanı kullanmalıyız.

izmhosp.dat

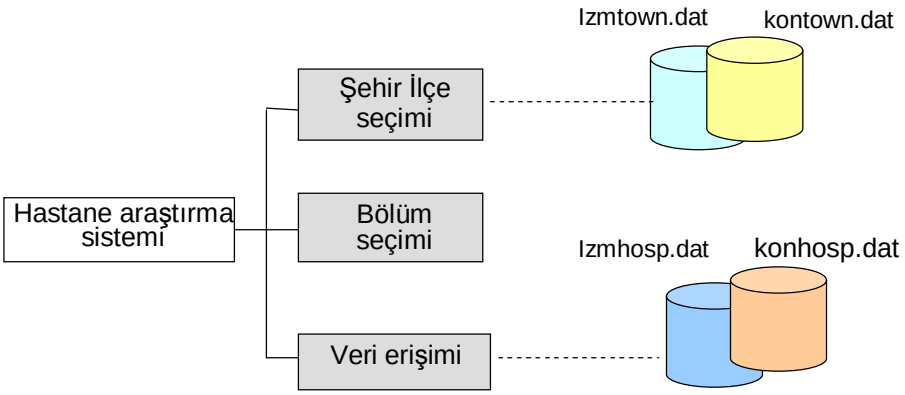
```
1 ABC Hastane   1379Sokak    1 1 0 0
1 ZZZ Klinik      1234Sokak     1 1 0 1
2 CCC Hastane   547Sokak      0 1 1 1
  :
```

Alanların içeriği

No	Nesne	Tipi	Açıklama
1	İlçe kodu	İnt	
2	Hastane adı1	string	
3	Hastane adı2	string	
4	Hastane adresi	string	
5	Dahiliye Bölümü için işaret	int	0--hayır 1--evet
6	Travma Cerrahisi için işaret	int	0--hayır 1--evet
7	Pediatri Bölümü için işaret	int	0--hayır 1--evet
8	Doğum kliniği için işaret	int	0--hayır 1--evet

b) Program Bileşenleri

Uygulamamızda büyük bir sistem oluşturmak istemediğimiz için bu sistem giriş yapmak ve dosyadaki veriyi düzenlemek için bir programa sahip değildir. Bunlar editör yardımıyla oluşturulacaktır. Buna göre program tek parça olabilir ve aşağıdaki gibi üçe bölünebilir.



Şekil 3.8:Hastane araştırma sisteminin program bileşenleri

Ø (4) Program Tasarımı

Program tasarımının örneği aşağıda gösterilmektedir. Dahili tasarımda üç bölüm görebilirsiniz.

Genel akış diyagramı

Değişken tablosu

Nesne	tipi	Kullanımı
dep_name[4][21]	char	Bölüm isimleri
town_name[20][20]	char	İlçe isimleri
city_sel	int	Şehir numarası seçimi
town_sel	int	İlçe numarası seçimi
div_sel	int	Bölüm numarası seçimi
h_town	int	İlçenin hastane sayısı
h_name1[10]	char	Hastane adı 1
h_name2[10]	char	Hastane adı 2
h_address[20]	char	hastane adresi
dep_f[4]	int	Bölüm işareti (0—dahiliye böl, . . .)

Değişken tablosu sadece önemli değişkenleri göstermektedir Bu değişkenlerin benzerleri burada listelenmemiştir.

Burada tekrar arama yapmak için programın bir kez çalışmasından sonra, döngü içinde şart kontrolü için do-while yapısı kullanılır. Bu bölümün kodlaması sağda gösterilmiştir.

Burada tekrar arama yapmak için programın bir kez çalışmasından sonra, döngü içinde şart kontrolü için do-while yapısı kullanılır. Bu bölümün kodlaması aşağıda gösterilmiştir.

```
do
{
        :

      printf("\nTekrar arama( 1--evet/ 2--hayır) ==>");
      scanf("%d",&retry);
}while (retry == 1);
```

Ø (5) Genel Program Parçaları

Aşağıdaki program genel akış diyagramının parçaları olan “Şehir ilçe seçimi” ve “Bölüm seçimi” içindir. Programda alt çizgi ile gösterilmiş olan boşlukları doldurunuz.

```
/*  Şehir – ilçe seçimi  */
/*  Bölüm seçimi  */

char dep_name[4][21]={"Dahiliye Bolumu","Travma Cerrahisi","Pediatri Bolumu",
                      "Doğum Klinigi"};
char town_name[20][18];
int town_num[20];
int city_sel,town_sel,dep_sel;
  :
/* Şehir seçimi ve gerekli dosyanın açılması */

  printf("\nSehir seciniz\n");
  printf("  1--Izmir   2--Konya\n");
  printf("    ===>");
  scanf("%d",&city_sel);
  if ( _____ )
  {
      fp1=fopen("izmtown.dat","r");
      fp2=fopen("izmhosp.dat","r");
  }
  else
  {
      fp1=fopen("kontown.dat","r");
      fp2=fopen("konhosp.dat","r");
  }
```

```
/* İlçe secimi */

   printf("\nİlce seciniz\n");
   i=1;
   rt = fscanf( _____ ,"%d%s",&town_num[i],town_name[i]);
    while (rt != _____ )
   {
      printf("%2d %-17s", _____ );
      i++;
      rt = fscanf( _____ );
   }
   fclose(fp1);
   printf("\n    ===>");
   scanf("%d",&town_sel);

   /* Bölüm secimi */
   printf("\n bolum seciniz\n");
   for( i= 0 ;_____;i++)
      printf("%2d %-20s\n",_____,dep_name[i]);
      printf("    ===>");
      scanf("%d",&dep_sel);
```

Aşağıdaki program genel akış diyagramının "Veri erişim" bölümü içindir. Programın veri kontrolünü nasıl yaptığını göz önünde tutarak alt çizgi ile verilmiş boşlukları tamamlayınız.

```
/* Veri girişi*/

    printf("\n %s  %s deki hastaneler asagıda gosterilmistir. \n",
    dep_name[dep_sel-1],town_name[town_sel]);
    printf("   Hastane adresleri\n");
    rt=fscanf(fp2,"%d%s%s%s%d%d%d%d",
    &town_f,h_name1,h_name2,h_address,&dep_f[0],&dep_f[1],&dep_f[2],&dep_f[3]);
    while (rt != EOF)
    {
        if( town_f ==_____ && dep_f[_____] == 1)
                printf("    %-10s%-10s     %-s\n",h_name1,h_name2,h_address);
                rt=fscanf(fp2,"%d%s%s%s%d%d%d%d",
        &town_f,h_name1,h_name2,h_address,&dep_f[0],&dep_f[1],&dep_f[2],&dep_f[3]
);
    }
```

UYGULAMA FAALİYETİ

Aşağıdaki veri erişim sistemlerini konu alan uygulama faaliyetini yapınız.

- Öğrenme faaliyetinde çalıştığınız hastane araştırma sistemini oluşturunuz. Dosyadaki veri en az, fakat sistemin kontrolü için yeterli olmalıdır.
- Veri erişimi ile ilişkili bir tema seçiniz ve kendi fikirlerinizden bir sistem oluşturunuz.

Erişim sistemi için örnekler

- Restoran araştırma
- Kiralık ev araştırma
- Otobüs araştırma (ücret , zaman….)
- Eşya araştırma
- Konser ve oyun araştırma

İşlem Basamakları	Öneriler
Ø Temel tasarım şemasını oluşturunuz. Ø Harici tasarım yapısını oluşturunuz. Ø Dahili tasarım yapısını oluşturunuz. Ø Program tasarımını yapınız. Ø Programda hata var ise bunları gideriniz. Ø Ekran görüntüsünü kontrol ediniz.	Ø İlk olarak ne yapmak istediğimize ve sistemin içeriğine karar veriniz. Ø Programın ekran görüntüsünü tasarlayınız. Ø Ekran görüntüsüne uygun olarak programda giriş ve çıkışların nasıl olacağına karar veriniz. Ø Sistemde kullanacağınız program ve dosya sayısını belirleyiniz. Ø Dosya isimlerinin dosya içeriğini hatırlatmasına özen gösteriniz. Ø Değişken tablosunu hazırlayınız. Ø Genel akış diyagramını çiziniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki ifadeleri doğru [D] veya yanlış [Y] olarak değerlendiriniz.

[ ] 1. Bir dosyayı açmak için **fopen** fonksiyonu kullanılır.

[ ] 2. **fopen(“personel”,”r”);** /* dosyayı okuma modunda açar.

[ ] 3. Dosya yazma modunda açılmışsa dosyanın eski içeriği kaybolmaz.

[ ] 4. **FILE *di** /* ifadesi dosya işaretçisini tanımlar. */

[ ] 5. **fclose** /* ifadesi ile dosya kapanır */

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modülde yaptığınız uygulamaları tekrar yapınız. Yaptığınız bu uygulamaları aşağıdaki tabloya göre değerlendiriniz.

AÇIKLAMA: Aşağıda listelenen kriterleri uyguladıysanız EVET sütununa, uygulamadıysanız HAYIR sütununa X işareti yazınız.		
Değerlendirme Ölçütleri	**Evet**	**Hayır**
1. include komutu ile istenilen kütüphaneleri belirttiniz mi?		
2. Değişken tanımlaması yaptınız mı?		
3. Dizi tanımlamasını yaptınız mı?		
4. Karakter katarı tanımlamasını yaptınız mı?		
5. scanf() fonksiyonu ile klavyeden dizi verilerini okuttunuz mu?		
6. scanf() fonksiyonu ile klavyeden karakterleri okuttunuz mu?		
7. printf() fonksiyonu ile dizi verilerini yazdınız mı?		
8. printf() fonksiyonu ile karakterleri yazdınız mı?		
9. Dizilerde sıralama işlemini yaptınız mı?		
10. Diziye veri ekleme işlemini yaptınız mı?		
11. Diziden veri çıkarma işlemini yaptınız mı?		
12. Karakter katarının kopyalama işlemini yaptınız mı?		
13. Karakter katarlarını karşılaştırdızı mı?		
14. Karakter katarlarını birbirine eklediniz mi?		
15. Karakter katarının uzunluğunu buldunuz mu?		
16. Dosyadan veri okuttunuz mu?		
17. Dosyaya veri yazdınız mı?		
18. Dosyaya veri ekledizi mi?		
19. Program yazım işlemlerinin doğruluğunu kontrol ettiniz mi?		

DEĞERLENDİRME

Hayır cevaplarınız var ise ilgili uygulama faaliyetini tekrar ediniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALIYETI 1 CEVAP ANAHTARI

1	**D**
2	**D**
3	**Y**
4	**Y**
5	**Y**
6	**D**
7	**D**
8	**D**
9	**D**
10	**Y**

ÖĞRENME FAALİYETİ 2 CEVAP ANAHTARI

1	**D**
2	**Y**
3	**D**
4	**Y**
5	**Y**
6	**Y**
7	**D**
8	**D**
9	**D**
10	**D**

ÖĞRENME FAALİYETİ 3 CEVAP ANAHTARI

1	**D**
2	**D**
3	**Y**
4	**D**
5	**Y**

KAYNAKÇA

- ISHIDA Yasuhiro, Murakamı Hideki, Koıchı Ito, Gürcan Çayır, **Bilgisayar Kontrol Teknolojisi,** M.E.B - JICA ,Eylül ,2005.