6. DİZİLER (ARRAYS)

1. Giriş

Bellekte ard arda sıralanan aynı türde değişken öbeğidir.

liste[0]	61
liste[1]	37
liste[2]	5
liste[3]	73
liste[4]	67
liste[5]	42

2. Tanımlanması

tür isim[adet];
int liste[6]; => 6 adet tamsayıdan oluşan dizi.
int liste[6]={61,37,5,73,67,42};
int liste[]={61,37,5,73,67,42};
int liste[6]={0}; => {0,0,0,0,0,0};
int liste[6]={1}; => {1,0,0,0,0,0};

int liste[]; OLMAZ! Eğer baştan bir şeye eşitlemiyorsan mutlaka boyutunu(adedini) belirtmelisin.
int liste[3]={3,34,57,4,2,6}; OLMAZ! Boyuttan fazla içerik giremezsin.

3. Erişimi

a. Sıradan Erişim

isim[indis] ya da isim[sira-1] şeklinde erişiriz.
Bilgisayarlar saymaya 0'dan başladığı için bir dizinin ilk elemanının indisi 0'dır. Dolayısıyla sıradan daima 1 çıkarmamız gerekir.

Buna Göre,
6 Elemanlı bir dizi olan liste'nin
ilk elemanı liste[0]
son elemanı liste[5] olur.

liste[3+2] ile liste[5] aynı anlama gelir, yani köşeli parantezler içerisinde istenilen işlem yapılabilir, liste[i*j] gibi başka değişkenleri de çarpabiliriz mesela.

Mesela,
/*10 elemanlı dizinin içindekileri döngünün her turunda a'ya aktarıyor ve a'nın yeni değerini ekrana yazdırıyor. */

```
int i, dizi[10];
for (i=0;i<10;i++){
        a=dizi[i];
        printf("a'nın yeni degeri: %d\n",a);
}
```

veya

```
/*10 elemanlı dizinin elemanlarını indisin iki katı, yani
0,2,4,6,8,10,12,14,16,18 yapar.*/

for (i=0;i<10;i++)
        dizi[i]=i*2;
```

b. Girdi/Çıktı Erişimi

Dizi elemanları normal değişkenler oldukları için onlara da printf ve scanf yoluyla erişebiliriz.

Örnek:

```
printf(“4. Elemanın değeri: %d”,c[3]);
printf(“5. Elemanın değerini giriniz:”);
scanf(“%d”,&c[4]);
/* c dizisinin elemanları normal değişken olduğu için değişkenin başına & işaretini koymayı unutmayalım.*/
```

c. Tablo ile Gösterim

```
#include <stdio.h>
int main(){
        int i, k[8];
        for (i=0;i<8;i++)
                printf(“%d\t”,k[i]);

        return 0;
}
```

4. Fonksiyonlar ve Diziler

Fonksiyonlar normal türde değişkenleri parametre olarak aldıkları gibi dizileri de parametre olarak almaktadırlar. Ancak burada durum biraz daha farklıdır. Normal bir fonksiyona bakalım:

```
#include <stdio.h>
int main(){
        int sayi=4;
        printf(“%d sayisinin iki kati %d'dir”,sayi,iki_katini_soyle(sayi));
        return 0;
}

int iki_katini_soyle(int x){
        int cevap;
        cevap=x*2;
        return cevap;
}
```

Burada printf'in içinde iki_katini_soyle fonksiyonuna değeri 4 olan sayı değişkenimizi gönderiyoruz. Bu 4 değeri fonksiyondaki x değişkenine yazılıyor. Yani fonksiyon çalışmaya başladığında x'in değeri 4. İçeride cevap 8 olarak hesaplanıyor ve cevap geri döndürülüyor. Döndürme işlemi yapılır yapılmaz ise fonksiyonla işimiz bittiği için hem x hem de cevap değişkenlerine ayrılan bellek alanları başka programlar kullansın diye serbest bırakılıyor. Dolayısıyla x ve cevap yok olmuş oluyorlar. Peki main'in içindeki sayi değişkeninin değeri değişti mi? Hayır! Çünkü sayi'nin değerinin kopyası x'e yazıldı, işi görüldükten sonra kopya yok edildi, sayiya bir zarar gelmedi. Bu tür fonksiyon çağırmaya “değerle çağırma(call by value)” denir.

Dizileri parametre olarak bir fonksiyona gönderirken ise durum farklıdır. Dizilerde eleman sayısı çok olduğu için, kolaylık açısından dizinin adresi ve dizinin boyutu(eleman sayısı) parametre olarak alınır. Böylece ihtiyacı olan fonksiyonun o adrese giderek belirtilen sayıda okuma yapması sağlanır. Örnek:

```
#include <stdio.h>
int main(){
	int liste[4]={3,4,7,8};

	printf("Liste = %d - %d - %d -%d",liste[0],liste[1],liste[2],liste[3]);

	iki_katini_al(liste,4); /*Dizinin ismini ("liste") yazdık, [] yok! */

	printf("Liste = %d - %d - %d -%d",liste[0],liste[1],liste[2],liste[3]);

	return 0;
}


void iki_katini_al(int gelenDizi[], int boyut){ /* Dikkat, gelenDizi[] yazdık*/
	int i;
	for (i=0;i<boyut;i++)
		gelenDizi[i]=gelenDizi[i]*2;

}
```

Burada gördüğünüz fonksiyon, bir tamsayı dizisini ve onun boyutunu parametre olarak almaktadır. İçeride ise elemanlarını tarayarak her birini alıp, ikiyle çarpıp kendi yerine koymaktadır. Bu işlemin sonucunda ise önceki örnekten farklı olarak liste adlı dizimizin özgün hali de değişmiş olur. Çünkü daha önce bahsettiğimiz üzere, int gelenDizi[] parametresi "liste" dizisinin adresini temsil etmektedir. Fonksiyon içerisinde ise belirtilen adresteki değeri alıp, yine o adrese değerin iki katı yazılmaktadır.

Bu tür fonksiyon çağrımına da "Adresle çağırma(call by reference)" denmektedir ve dizilerin özgün hallerinde değişiklik yaptığı için tehlikeli sonuçlara yol açabilmektedir. Eğer özgün hallerinin değiştirilmesini istemiyor, sadece okunmasını istiyorsak kodlama yaparken bu hataya düşmemek için güvenlik önlemi olarak int'in önüne const anahtarını koyarız. Bu anahtar, fonksiyon içinde belirtilen dizinin değiştirilmemesini sağlar. Değiştirmeye kalksanız bile derleyici hata verir.

İmla olarak dikkat etmeniz gereken bir husus ise, fonksiyon parametrelerinde alınacak parametrenin bir dizi olduğunu belirtmek için yerel adın (gelenDizi) sonuna [] işaretleri koyduk. Bu işaretler gelenin dizi olduğunu belirtmek için mutlaka gerekmektedir. Bunun yanında bu köşeli parantezlerin içine herhangi bir sayı yazılmaz, dizinin boyutu ayrı bir parametre olarak gönderilir. gelenDizi[] bir diziyi ifade ederken gelenDizi[3] ise henüz oluşturulmamış bir dizinin(gelenDizi) 4. elemanını ifade etmektedir.

Fonksiyonları çağırırken ise dizinin sadece ismi kullanılır. Burada [] kullanılmaz.

void iki_katini_al(int gelenDizi[], int boyut) /*gelenDizi[] kullandık!*/
iki_katini_al(liste,4); /* ”liste” kullandık. [] YOK!

Önceki sayfadaki program kodunun çıktılarına bakacak olursak:

Liste = 3, 4, 7, 8
Liste = 6, 8, 14, 16

basacaktır. Çünkü listenin özgün hali değişmiştir.

5. Dizi Uygulamaları

a. Yer Değiştirme

Bir dizinin iki elemanının yerini değiştirmek istiyorsak, geçici bir değişken tanımlamalıyız.

```
int gecici;
gecici=dizi[0];
dizi[0]=dizi[1];
dizi[1]=gecici;
```

dersek, dizi'nin 0 ve 1 indisli elemanları yer değiştirmiş olur.

b. Baloncuk/Kabarcık Sıralaması

Bu sıralama, sayılardan oluşan bir dizinin küçükten büyüğe(veya tersi) sıralanması için kullanılan kullanışlı yöntemlerden birisidir. Mantığı, diziyi baştan sona tarayarak ardışık iki elemanı karşılaştırmak, eğer birincisi ikincisinden daha büyük ise bunların yerini değiştirmek ve bir sonraki elemanı iki sonraki elemanla karşılaştırmak ve bunu sondan 2. elemana kadar sürdürmektir. Sondan 2. eleman, çünkü son elemana gelmiş olsak karşılaştıracağımız bir sonraki eleman olmazdı.

Tüm bu tarama işlemi ise dizinin eleman sayısınca yapılır. Çünkü tek bir turda gözden kaçan şeyler olacağı için sıralama tam olarak yapılmış olmaz.

```
#include <stdio.h>

int main(){

        int a[10]={2,6,4,8,10,12,89,68,45,37};
        int i, tur, gecici;

        printf(“Duz siradaki hali:);
        for (i=0;i<10;i++)
                printf(“%d, “,a[i]);
```

```
        for (tur=0;tur<10;tur++){
                for (i=0;i<9;i++){ /*Dikkat ederseniz en son 8. elemana
                                                geliyor*/
                        if (a[i]>a[i+1]){
                                gecici=a[i];
                                a[i]=a[i+1];
                                a[i+1]=gecici;
                        }

                }
        }

        printf("Siralanmis hali:);
        for (i=0;i<10;i++)
                printf("%d, ",a[i]);

        return 0;
}
```

c. Doğrusal(Linear) Arama

Dizilerin içinde aradığımız değerin nerede olduğunu bulmak için kullanılan çeşitli yöntemlerden birisidir. Mantığı, tüm diziyi tarayarak her bir elemanın aranan ile aynı olup olmadığının denetlenmesidir.

```
#include <stdio.h>

int main(){
 int liste[10]={1,4,3,6,5,8,54,93,58,34};
 int i,aranan;
 printf("Aranacak sayiyi girin:");
 scanf("%d",&aranan);

 for (i=0;i<10;i++){
 if (liste[i]==aranan)
 printf("%d indisinde,%d. sirada %d buldum",i,i+1,aranan);
 }

 return 0;
}
```

d. İkili(Binary) Arama

Bu arama yöntemi doğrusal aramaya göre çok daha hızlıdır. Ancak kullanılabilmesi için dizinin sıralanmış olması gerekmektedir. Mantığı, dizinin ortasındaki elemandan başlayıp, aranan daha küçükse sola(düşük indislere), daha büyükse sağa(yüksek indislere) yönelmesidir.

```
#include <stdio.h>

int ikiliArama(int liste[], int aranan, int alt_sinir,int ust_sinir);

int main(){
	int listem[10]={1,3,4,6,8,9,11,15,16,80}, sayi, sonuc;
	printf("Hangi sayiyi arayacaksiniz?");
	scanf("%d",&sayi);
	sonuc = ikiliArama(listem, sayi,0,9);
	if (sonuc == -1){
		printf("%d bulunamadi!\n",sayi);
		return 0;
	}

	printf("%d, %d indisinde bulundu.\n", sayi, sonuc );

	return 0;
}

int ikiliArama(int liste[], int aranan, int alt_sinir, int ust_sinir){
	int orta;

	if (ust_sinir < alt_sinir)
		return -1;

	orta=(alt_sinir+ust_sinir)/2;
	if (aranan > liste[orta])
		return ikiliArama(liste, aranan, orta+1, ust_sinir);
	else if (aranan < liste[orta])
		return ikiliArama(liste, aranan, alt_sinir, orta-1);
	else
		return orta;

}
```

6. KARAKTER DİZİLERİ (STRINGS)

Sayılardan bir dizi oluşturabildiğimiz gibi karakterlerden de dizi oluşturabiliriz. Bununla birlikte bu dizileri birer metinmiş veya cümleymiş gibi kullanabiliriz. Örneğin,

```
char metin[]={'M','e','r','h','a','b','a','\0'};
```

Burada dikkat etmemiz gereken şey, her karakter dizisinin sonuna \0 (NULL / Sonlandırma[Terminating]) karakterinin gelmesi gerektiğidir. Bunun sebebi ise şudur: Printf veya puts gibi metin basan fonksiyonlar, nerede duracaklarını bilemezler. Eğer \0 olmasaydı, h,a,b,a harflerini geçip hafızada alakasız diğer alanları da basmaya kalkacaktı. Halbuki bizim metnimizin Merhaba'dan hemen sonra bittiğini belirtmemiz gerek.

Bu yüzden her stringin sonuna sonlandırma karakteri koyuyoruz.

String Tanımlamak

```
char metin[]={'M','e','r','h','a','b','a','\0'}; //Boyutu belirtmeden
char metin[8]={'M','e','r','h','a','b','a','\0'}; //Boyutu belirterek (7+1)
char metin2[]="Merhaba"; //Tırnak içinde yazınca \0 koymamalıyız.
char metin2[8]="Merhaba"; //Sayıya \0'ı da katıyoruz. (7+1)
char metin2[10]="Selam"; //10 Hane ayırıp 6'sını doldurabiliriz de.

printf("Bir metin giriniz:");
scanf("%s",metin); //%s=string, metin ise karakter dizisinin ismi.[ ] yok!

printf("Bir metin giriniz:");
gets(metin); //gets doğrudan string olarak almaya yarıyor.

int i=0;
do {
    harf=getchar();
    metin[i++]=harf;
    } while (harf!='\0')
```

String Kullanımı

```
int i;
for (i=0;i<10;i++){
        if (metin[i]==arananHarf){
                printf("Buldum: %d.indis",i);
                break;
        }
```

```
int i=0;
while (metin[i++]!=arananHarf); // ; var. Sadece aradığım harfe getiriyor.
   printf("Aradigim harfi buldum : %d. indis",i-1); //i++ fazladan 1 arttırdı.
```

```
printf("Kacinci harfi degistireceksin?");
scanf("%d",&kacinci);

printf("Hangi harfi koyacaksin?");
scanf("%c",&harf);

metin[kacinci-1]=harf; //indis=sira-1 olduğu için -1 var.
```

Özgür İşletim Sistemi PARDUS üzerinde OpenOffice ile hazırlanmıştır.

Emre Aladağ