Bir C programı içerisinde, dizilerin boyutu ve kaç elemanlı olduğu program başında belirtilirse, derleyici o dizi için gereken bellek alanını (bölgesini) program sonlanıncaya kadar saklı tutar ve bu alan başka bir amaç için kullanılamaz [1]. Bu türden diziler **statik dizi** olarak isimlendirilir. Statik dizinin boyutu programın çalışması esnasında (run time) değiştirilemez.

Fakat, programın çalışırken bir dizinin boyutu ve eleman sayısı bazı yöntemler kullanılarak değiştirilebilir. Bu tür dizilere **dinamik dizi** denir. Dinamik diziler için gereken bellek bölgesi, derleyici tarafından işletim sisteminden istenir, kullanılır ve daha sonra istenirse bu bölge boşaltırılır. Bu bölümde, dinamik dizi kavramı ve dinamik bellek yönetimi anlatılacaktır.

13.1 Dinamik Dizi Fonksiyonları

ANSI C'de, dinamik diziler işaretçi kullanılarak ve standart kütüphanedeki `malloc()`, `calloc()`, `realloc()` ve `free()` fonksiyonlarının yardımıyla ile oluşturulur veya boşaltılır. Bu fonksiyonlar Tablo 13.1 de listelenmiştir.

Tablo 13.1: *`stdlib.h` kütüphanesindeki dinamik bellek fonksiyonları*

Dinamik Bellek Fonksiyonu	**Açıklama**
`void *malloc(size_t eleman_sayısı);`	Bellekte herbiri `size_t` tipinde olan *`eleman_sayısı`* kadar yer (bellek bloğu) ayırır. Bu yer verilmezse geriye `NULL` gönderir.
`void *calloc(size_t eleman_sayısı, size_t nbayt);`	Bellekte herbiri `nbayt` kadar yer işgal edecek *`eleman_sayısı`* kadar boş yer ayırır ve bütün bitleri sıfırlar. Bu yer ayrılamazsa geriye `NULL` gönderir.
`void *realloc(void *ptr, size_t nbayt);`	*`ptr`* işaretçisi ile gösterilen bellek bloğunu, *`nbayt`* kadar büyüterek veya küçülterek değiştirir. Bu iş gerçekleşmezse geriye `NULL` gönderir.
`void free(void *ptr);`	Daha önce ayrılan adresi *`ptr`*'de saklanan bellek alanının boşaltır.

Tamsayı tipinde bir dinamik dizi tanımlanırken aşağıdaki işlem basamakları izlenmelidir:

```
/* dinamik dizi bildirimi */
int *dizi;

/* eleman saysını belirle */
scanf("%d",&n);

/* n tane bellek bloğu isteniyor */
dizi = (int *) malloc( sizeof(int)*n );
```

```
/* Boş yer varmı sorgulanıyor */
if( dizi == NULL )
   printf("Yetersiz bellek alanı\n"), exit(1);

...
/* dizi burada kullanılıyor */
...

/* bellek bloğu boşaltılıyor */
free(dizi);
```

Program 13.1, eleman sayısı klavyeden girilen bir dizinin aritmetik ortalamasını hesaplar. Eleman sayısı sıfır veya negatif bir değer olduğunda, sonsuz döngüden çıkılır ve program sonlanır. İnceleyiniz.

Program 13.1: *Dinamik dizi ile ortalama hesabı*

```
01: /* 13prg01.c: Dinamik dizi ile ortalama hesabı */
02:
03: #include <stdio.h>
04: #include <stdlib.h>
05:
06: int main(){
07:
08:   int n,i;
09:   float *x, toplam, ort;
10:
11:   while(1)
12:   {
13:         /* dizinin eleman sayısı okunuyor */
14:         printf("\nEleman sayısını girin: ");
15:         scanf("%d",&n);
16:
17:         /* eleman sayısı <= 0 ise dögüden çık */
18:         if( n<=0 )
19:           break;
20:
21:         /* bellekten yer isteniyor */
22:         x = (float *)  malloc( sizeof(float)*n );
23:
24:        /* istenen yer ayrıldı mı? */
25:        if( x == NULL ){
26:          puts("Yetersiz bellek alanı");
27:          exit(1);
28:        }
29:
30:        /* elemanlar tek tek belleğe yazılıp
31: toplamları hesaplanıyor */
32:        for(toplam =0.0, i=0; i<n; i++){
33:           printf("%d. eleman: ",i+1);
34:           scanf("%f",&x[i]);
35:           toplam += x[i];
36:        }
37:
38:        ort = toplam / n;
39:
40:        printf("Ortalama = %f\n",ort);
```

```
41:
42:         /* ayrılan alan boşaltılıyor */
43:         free(x);
44:    }
45:
46:  return 0;
     }
```

ÇIKTI

```
Eleman sayısını girin: 2
1. eleman: 4
2. eleman: 6
Ortalama = 5.000000

Eleman sayısını girin: 4
1. eleman: 2
2. eleman: 3
3. eleman: 1
4. eleman: 5
Ortalama = 2.750000

Eleman sayısını girin: 0
```

Karakter dizileri içinde benzer adımlar uygulanır. Program 13.2'de tanımlanan `altKatar()` fonksiyonu, bir katarın alt parçalarını geri döndürür.

Program 13.2: *Bir karakter dizisinin alt parçalarının bulunması*

```
01: /* 13prg02.c: Bir katarın parçalarini (alt katar)
02: veren fonksiyon */
03:
04: #include <stdio.h>
05: #include <stdlib.h>
06:
07: char *altKatar(char *str, int, int);
08:
09: int main()
10: {
11:    int  i;
12:    char *s, *parca;
13:
14:    s = "programlama";
15:
16:    for(i=0; s[i]; i++)
17:    {
18:       parca = altKatar(s, 0, i);
19:       puts(parca);
20:    }
21:
22:  return 0;
23: }
24:
25: /* str'nin p1. elemanindan p2. elemanina kadar olan
26: alt katarını gonderir. */
27: char *altKatar(char *str, int p1, int p2)
28: {
29:    int i, j=0, n;
30:    static char *alt;
```

```
31:     n = p2 - p1;
32:
33:     /* bellekten yer ayrılıyor... */
34:     alt = (char *) malloc( n*sizeof(char) );
35:
36:     for(i=p1; i<=p2; i++)
37:       alt[j++] = str[i];
38:
        return alt;
    }
```

ÇIKTI

```
p
pr
pro
prog
progr
progra
program
programl
programla
programlam
programlama
```

13.2 Dinamik Matrisler

İki veya daha fazla boyuta sahip dinamik dizi oluşturmak mümkündür. Bu durumda, göstericiyi gösteren göstericler kullanılır. Program 13.3, bir matrisin elemanları bulup ekrana yazar.

Program 13.3: *Dinamik matris tanımlama*

```
01: /* 13prg02.c: Dinamik matris tanımlama */
02:
03: #include <stdio.h>
04: #include <stdlib.h>
05:
06: int main()
07: {
08:     int **matris;
09:     int satir, kolon;
10:     int s, k;
11:     int i;
12:
13:     printf("Matrisin satır sayısı: ");
14:     scanf("%d", &satir);
15:
16:     printf("Matrisin kolon sayısı: ");
17:     scanf("%d", &kolon);
18:
19:     /* dıştaki dizi için bellek alanı isteniyor  */
20:     matris = (int **) calloc(satir, sizeof(int));
21:
22:     /* içteki dizi için bellek alanı isteniyor */
23:     for(i = 0; i < satir; i++)
24:         matris[i] = (int *) calloc(kolon,
25: sizeof(int));
```

```
26:
27:      /* matrisin elemanları okunuyor */
28:      for(s = 0; s < satir; s++)
29:          for(k = 0; k < kolon; k++) {
30:              printf("Matrisin elemanı girin:
31:  matris[%d][%d] = ", s, k);
32:              scanf("%d", &(matris[s][k]));
33:          }
34:
35:      printf("\nGirilen matris:\n");
36:
37:      for(s = 0; s < satir; s++) {
38:          for(k = 0; k < kolon; k++)
39:              printf("%4d", matris[s][k]);
40:
41:          printf("\n");
42:      }
43:
44:      /* içteki dizi boşaltılıyor */
45:      for(i = 0; i < satir; i++)
46:          free((void *) matris[i]);
47:
48:      /* dıştaki dizi boşaltılıyor */
49:      free((void *) matris);
50:
         return(0);
     }
```

ÇIKTI

```
Matrisin satır sayısı: 2
Matrisin kolon sayısı: 3
Matrisin elemanı girin: matris[0][0] = 1
Matrisin elemanı girin: matris[0][1] = 2
Matrisin elemanı girin: matris[0][2] = 3
Matrisin elemanı girin: matris[1][0] = 5
Matrisin elemanı girin: matris[1][1] = 6
Matrisin elemanı girin: matris[1][2] = 8

Girilen matris:
   1   2   3
   5   6   8
```