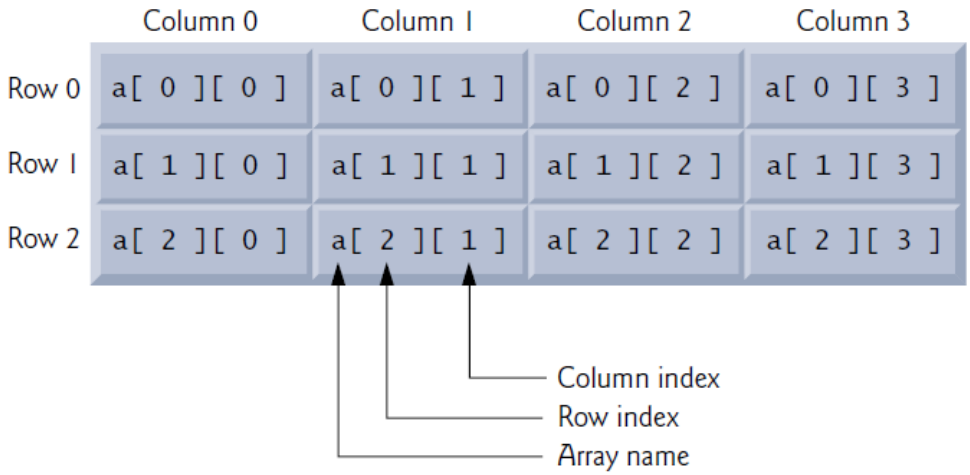
Çok Boyutlu Diziler

C'deki diziler birden çok dizine sahip olabilir. C standardının çok boyutlu diziler olarak atıfta bulunduğu çok boyutlu dizilerin yaygın bir kullanımı, satırlar ve sütunlar halinde düzenlenmiş bilgilerden oluşan değer tablolarını temsil etmektir. Belirli bir tablo öğesini tanımlamak için iki dizin belirtmeliyiz: İlki öğenin satırını ve ikincisi öğenin sütununu tanımlar. Belirli bir öğeyi tanımlamak için iki dizin gerektiren tablolara veya dizilere iki boyutlu diziler denir.

Şekil 6.20 iki boyutlu bir diziyi göstermektedir, a. Dizi üç satır ve dört sütun içerir, bu nedenle 3'e 4 dizi olduğu söylenir. Genel olarak, m satır ve n sütun içeren bir diziye mxn dizisi denir.



Şekil 6.20; Üç satır ve dört sütundan oluşan iki boyutlu dizi.

“a” dizisindeki her öğe, Şekil 6.20'de a[i][j] biçiminde bir öğe adıyla tanımlanır; a, dizinin adıdır ve i ve j, a'daki her bir öğeyi tanımlayan dizinlerdir. 0 satırındaki öğelerin adlarının hepsinin ilk dizini 0'dır; 3. sütundaki öğelerin adlarının hepsinin ikinci indeksi 3 olur, … gibi.

Uygun tanımlandığında çok boyutlu bir dizi başlatılabilir. Örneğin, iki boyutlu bir dizi int b[2][2] ile tanımlanabilir ve başlatılabilir;

int b[2][2] = {{1, 2}, {3, 4}};

Değerler parantez içinde satırlara göre gruplandırılmıştır. İlk parantez kümesindeki değerler 0. satırı başlatır ve ikinci parantez kümesindeki değerler 1. satırı başlatır. Böylece, 1 ve 2 değerleri sırasıyla b[0][0] ve b[0][1] öğelerini başlatır. 3 ve 4 değerleri sırasıyla b[1][0] ve b[1][1] öğelerini başlatır. Belirli bir satır için yeterli başlatıcı yoksa, o satırın geri kalan öğeleri 0 olarak başlatılır. Örneğin;

int b[2][2] = {{1}, {3, 4}};

Bu durumda, b[0][0] ile 1, b[0][1] ile 0, b[1][0] ile 3 ve b[1][1] ile 4'ü başlatır. Şekil 6.21, iki boyutlu dizileri tanımlamayı ve başlatmayı göstermektedir.

Bu örnekte;2x3 boyutlu bir matrisin yazdırılması gösterilmektedir. Eğer matris indislerine denk gelen değerler boş bırakılırsa o değerler 0 alınır. Burada 3 farklı matris örneği yer almaktadır.

“array1” Dizisi; Program, iki satır ve üç sütundan oluşan üç dizi (her biri altı öğe) tanımlar. array1'in tanımı (satır 10), iki alt listede altı başlatıcı değer ile başlatılır. İlk alt liste, dizinin 0. satırını 1, 2 ve 3 değerleri ile başlatır; ve ikinci alt liste, dizinin 1. satırını 4, 5 ve 6 değerleri ile başlatır.

“array2” Dizisi; Her alt listenin etrafındaki ayraçlar “array1” başlatıcı listesinden kaldırılırsa, derleyici ilk satırın öğelerini ve ardından ikinci satırın öğelerini başlatır. array2'nin tanımı (satır 14), beş başlatıcı ile başlatılmıştır. Başlatıcılar ilk satıra, ardından ikinci satıra atanır. Açık bir başlatıcıya sahip olmayan tüm öğeler otomatik olarak sıfır ile başlatılır. Bu nedenle array2[1][2], 0 olarak başlatılır.

“array3” Dizisi; (satır 18), iki alt listede üç başlatıcı vardır. Birinci satırın alt listesi, ilk satırın ilk iki öğesini 1 ve 2 olarak başlatır. Üçüncü öğe sıfır olarak başlatılır. İkinci sıra için alt liste, ilk öğeyi 4 olarak başlatır. Son iki öğe sıfır olarak başlatılır.

```
1  // Fig. 6.21: fig06_21.c
2  // Initializing multidimensional arrays.
3  #include <stdio.h>
4
5  void printArray(int a[][3]); // function prototype
6
7  // function main begins program execution
8  int main(void)
9  {
10    int array1[2][3] = {{1, 2, 3}, {4, 5, 6}};
11    puts("Values in array1 by row are:");
12    printArray(array1);
13
14    int array2[2][3] = {1, 2, 3, 4, 5};
15    puts("Values in array2 by row are:");
16    printArray(array2);
17
18    int array3[2][3] = {{1, 2}, {4}};
19    puts("Values in array3 by row are:");
20    printArray(array3);
21 }
22
23 // function to output array with two rows and three columns
24 void printArray(int a[][3])
25 {
26    // loop through rows
27    for (size_t i = 0; i <= 1; ++i) {
28
29       // output column values
30       for (size_t j = 0; j <= 2; ++j) {
31          printf("%d ", a[i][j]);
32       }
33
34       printf("\n"); // start new line of output
35    }
36 }
```

Şimdi bu kodu derleyip çalıştıralım. Sonra matris elemanlarını değiştirerek sonuçları inceleyelim.

İki boyutlu dizilerde dizi elemanlarına istediğimiz değerleri döngü içerisinde atayabiliriz. Örneğin, aşağıdaki ifade, Şekil 6.20'deki a dizisinin 2. satırındaki tüm öğeleri sıfırlar: Döngü yalnızca ikinci (sütun) dizini değiştirir. Önceki for ifadesi, atama ifadelerine eşdeğerdir:

```
for (column = 0; column <= 3; ++column) {
a[2][column] = 0;
}
```

```
a[2][0] = 0;
a[2][1] = 0;
a[2][2] = 0;
a[2][3] = 0;
```

- Öğeleri iki boyutlu bir dizide toplamak için aşağıdaki iç içe geçmiş “for” ifadesi kullanılabilir.

```
total = 0;
for (row = 0; row <= 2; ++row) {
        for (column = 0; column <= 3; ++column) {
                total += a[row][column];
        }
}
```

“for” ifadesi, dizinin öğelerini her seferinde bir satır olarak toplar. Dış “for” ifadesi, satırı (yani satır dizini) 0'a ayarlayarak başlar, böylece bu satırın öğeleri iç “for” ifadesi tarafından toplanabilir. Dış “for” ifadesi daha sonra satırı 1'e yükseltir, böylece o satırın öğeleri toplanabilir. Ardından, dış “for” ifadesi satırı 2'ye yükseltir, böylece üçüncü satırın öğeleri toplanabilir. “for” döngüleri sona erdiğinde, “total” değişkeni, a dizisindeki tüm öğelerin toplamını içerir.

-İki Boyutlu Dizi İşlemleri:

Şekil 6.22, “for” deyimlerini kullanarak 3'e 4'lük bir “StudentGrades” dizisinde birkaç başka yaygın dizi işlemlerini gerçekleştirir. Dizinin her satırı bir öğrenciyi temsil eder ve her sütun öğrencilerin dönem boyunca girdiği dört sınavdan birindeki notu temsil eder. Dizi işlemleri dört fonksiyon tarafından gerçekleştirilir. Minimum fonksiyonu (39-56. satırlar), herhangi bir öğrencinin o dönem için aldığı en düşük notu belirler. Maksimum fonksiyonu (59-76. satırlar), herhangi bir öğrencinin o dönem için aldığı en yüksek notu belirler. Ortalama Fonksiyonu (satır 79–89), belirli bir öğrencinin dönem ortalamasını belirler. “printArray” fonksiyonu (satır 92–108), iki boyutlu diziyi tablo biçiminde yazdırır.

Şimdi Örnek 6.22’yi bölüm bölüm ele alalım. Bu bölümde fonksiyon prototipleri (minimum, maximum, average ve printArray) oluşturulmaktadır. Burada const(sabit) ifadesini kullanmamızın ve ilk köşeli parantezin boş olmasının sebebi yapacağımız işlemlerin öğrenciye özel olması. Mesela sadece 0. öğrencinin aldığı notların toplanması ve ortalamasının alınması, … gibi.

```
1  // Fig. 6.22: fig06_22.c
2  // Two-dimensional array manipulations.
3  #include <stdio.h>
4  #define STUDENTS 3
5  #define EXAMS 4
6
7  // function prototypes
8  int minimum(const int grades[][EXAMS], size_t pupils, size_t tests);
9  int maximum(const int grades[][EXAMS], size_t pupils, size_t tests);
10 double average(const int setOfGrades[], size_t tests);
11 void printArray(const int grades[][EXAMS], size_t pupils, size_t tests);
```

Bu bölümde ise ana fonksiyon verilmektedir. Ana fonksiyonda öğrenci notları matris şeklinde girilmiştir. Matrisin ilk satırı 0 ncı öğrencinin ikinci satırı 1nci öğrencinin, üçüncü satırı ise 2nci öğrencinin notlarını göstermektedir. 24ncü satırda öğrencilerin sınav verilerini average (ortalama alma) fonksiyonuna gönderir ve ortalamalarını teker teker yazdırır. 28 ve 29 ncu satırlarda, "Printf" içine konulan minimum ve maximum fonksiyonuna değer gönderme işlemi aşağıdaki gibi olur;

const int grades[ ]
[EXAMS]=studentGrades:
STUDENT=size_t pupils:
EXAMS=size_t tests

32nci satırda kullandığımız "for" döngüsü ile 0. 1. ve 2. öğrencilerin sınav verilerini average (ortalama alma) fonksiyonuna gönderir ve ortalamalarını teker teker yazdırır.

```
12
13 // function main begins program execution
14 int main(void)
15 {
16    // initialize student grades for three students (rows)
17    int studentGrades[STUDENTS][EXAMS] =
18       { { 77, 68, 86, 73 },
19         { 96, 87, 89, 78 },
20         { 70, 90, 86, 81 } };
21
22    // output array studentGrades
23    puts("The array is:");
24    printArray(studentGrades, STUDENTS, EXAMS);
25
26    // determine smallest and largest grade values
27    printf("\n\nLowest grade: %d\nHighest grade: %d\n",
28       minimum(studentGrades, STUDENTS, EXAMS),
29       maximum(studentGrades, STUDENTS, EXAMS));
30
31    // calculate average grade for each student
32    for (size_t student = 0; student < STUDENTS; ++student) {
33       printf("The average grade for student %u is %.2f\n",
34          student, average(studentGrades[student], EXAMS));
35    }
36 }
```

Bu bölümde ise minimum fonksiyonu verilmektedir. Ana programdan minimum fonksiyonu çağrıldığında program buraya atlar ve bu fonksiyonu çalıştırır. 41nci satırda lowGrade = 100 kullanmamızın sebebi 100 den başlayıp karşılaştırarak daha düşük değerleri "lowGrade" değişkenine atıp sınavda alınan en düşük nota ulaşmaktır.
44ncü satırda (pupils=STUDENT), 47nci satırda (test=EXAMS) değişkenlerine gitmektedir.
49 ncu satırda değer 100 den küçükse değeri lowGrade'e eşitle böylece sınavda alınan en küçük notu bulmuş oluruz. 55nci satırda bulduğumuz en küçük notu (return komutu ile) ana fonksiyona geri döndürüyoruz.

```
37
38  // Find the minimum grade
39  int minimum(const int grades[][EXAMS], size_t pupils, size_t tests)
40  {
41     int lowGrade = 100; // initialize to highest possible grade
42
43     // loop through rows of grades
44     for (size_t i = 0; i < pupils; ++i) {
45
46        // loop through columns of grades
47        for (size_t j = 0; j < tests; ++j) {
48
49           if (grades[i][j] < lowGrade) {
50              lowGrade = grades[i][j];
51           }
52        }
53     }
54
55     return lowGrade; // return minimum grade
56  }
```

Bu bölümde ise benzer şekilde maksimum fonksiyonu verilmektedir. Burada da maximum değeri bulup ana programa döndürüyoruz.

```
57
58  // Find the maximum grade
59  int maximum(const int grades[][EXAMS], size_t pupils, size_t tests)
60  {
61     int highGrade = 0; // initialize to lowest possible grade
62
63     // loop through rows of grades
64     for (size_t i = 0; i < pupils; ++i) {
65
66        // loop through columns of grades
67        for (size_t j = 0; j < tests; ++j) {
68
69           if (grades[i][j] > highGrade) {
70              highGrade = grades[i][j];
71           }
72        }
73     }
74
75     return highGrade; // return maximum grade
76  }
77
```

Bu bölümde ise ortalama hesaplayan fonksiyon verilmiştir. 84ncü satırda öğrencilerin notları toplanıp "total" değikenine aktarılıp "return" ile ana programa döndürülmektedir. Buradaki "set0fGrades[i]'in tek parantezli olmasının sebebi grades[ ][ ] ilk parantezin değerinin belli olması ve ikinci parantezde bir döngü işleminin yapılmak istemesidir. Çünkü "grades[ ][EXAMS]=setOfGrades[ ]" olacak şeklinde bir atama yaptık. Yani satırları zaten belirli ve biz her öğrenci için sütunlarda yer alan not değerlerini toplamak istiyoruz. Bulduğumuz son "total" değerini toplam test sayısına bölüp (ortalamasını alıp) ana fonksiyona geri döndürüyoruz.

```
78  // Determine the average grade for a particular student
79  double average(const int setOfGrades[], size_t tests)
80  {
81     int total = 0; // sum of test grades
82
83     // total all grades for one student
84     for (size_t i = 0; i < tests; ++i) {
85        total += setOfGrades[i];
86     }
87
88     return (double) total / tests; // average
89  }
```

Bu bölümde ise printArray fonksiyonu verilmiştir. Hesaplanan ana fonksiyona (programa) döndürülen değerler ile program bu fonksiyona atlayarak hesaplanan değerleri yazdırır.

```
90
91  // Print the array
92  void printArray(const int grades[][EXAMS], size_t pupils, size_t tests)
93  {
94     // output column heads
95     printf("%s", "                [0]  [1]  [2]  [3]");
96
97     // output grades in tabular format
98     for (size_t i = 0; i < pupils; ++i) {
99
100       // output label for row
101       printf("\nstudentGrades[%u] ", i);
102
103       // output grades for one student
104       for (size_t j = 0; j < tests; ++j) {
105          printf("%-5d", grades[i][j]);
106       }
107    }
108 }
```

Program çalıştığında ise aşağıdaki veriler yazdırılır.

```
The array is:
                [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

Minimum, maksimum ve printArray fonksiyonlarının her biri üç bağımsız değişken alır; studentGrades dizisi (her fonksiyonda notlar olarak adlandırılır), öğrenci sayısı (dizinin satırları) ve sınav sayısı (dizinin sütunları). Her fonksiyon, iç içe “for” ifadeleri kullanarak döngü yapar.

```
// loop through rows of grades
for (i = 0; i < pupils; ++i) {
   // loop through columns of grades
   for (j = 0; j < tests; ++j) {
      if (grades[i][j] < lowGrade) {
         lowGrade = grades[i][j];
      }
   }
}
```

Dış “for” ifadesi, i'yi (yani satır dizini) 0'a ayarlayarak başlar, böylece o satırın öğeleri (yani, ilk öğrencinin notları), iç “for” ifadesindeki “lowGrade değişkeniyle karşılaştırılabilir. İçteki “for” deyimi, belirli bir satırın dört derecesi arasında dolaşır ve her bir notu “lowGrade” ile karşılaştırır. Bir not “lowGrade”'den düşükse, “lowGrade” o nota ayarlanır. Dış “for” ifadesi daha sonra satır dizinini 1'e yükseltir. Bu satırın öğeleri, “lowGrade” değişkeniyle karşılaştırılır. Dış “for” ifadesi daha sonra satır dizinini 2'ye yükseltir. Bu satırın öğeleri “lowGrade” değişkeniyle karşılaştırılır. İfadenin yürütülmesi tamamlandığında, “lowGrade” iki boyutlu dizideki en küçük notu içerir. Maksimum işlev, minimum işleve benzer şekilde çalışır.

Ortalama işlevi (satır 79-89) iki bağımsız değişken alır—belirli bir öğrenci için “setOfGrades” adlı iki boyutlu test sonuçları dizisi ve dizideki test sonuçlarının sayısı. Ortalama çağrıldığında, ilk bağımsız değişkene “studentGrades[öğrenci]” iletilir. Bu, iki boyutlu dizinin bir satırının adresinin ortalamaya geçirilmesine neden olur. “studentGrades[1]” bağımsız değişkeni, dizinin 1. satırının başlangıç adresidir.

İki boyutlu bir dizinin temel olarak tek boyutlu dizilerden oluşan bir dizi olduğunu ve tek boyutlu bir dizinin adının dizinin bellekteki adresi olduğunu unutmayın. Ortalama işlevi, dizi öğelerinin toplamını hesaplar, toplamı test sonuçlarının sayısına böler ve kayan nokta sonucunu verir.