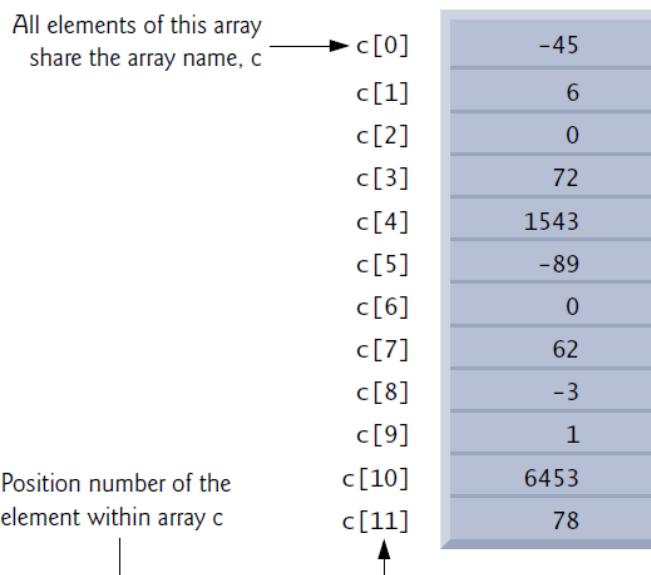


BİLGİSAYAR PROGRAMLAMA

DİZİLER ÖZET-1;

- Bir dizi, tümü aynı türe sahip olan bitişik bellek konumları grubudur.
- Dizideki belirli bir konuma veya öğeye atıfta bulunmak için, dizinin adını ve dizideki belirli ögenin konum numarasını belirtiriz.
- Bu öğelerden herhangi biri, dizinin adını ve ardından belirli ögenin konum numarasını köşeli parantezler ([]) içinde vererek anılabılır.
- Her dizideki ilk öğe, sıfırıncı öğedir (yani, konum numarası 0 olan). Bir dizi adı, diğer tanımlayıcılar gibi yalnızca harf, rakam ve alt çizgi içerebilir ve bir rakamla başlayamaz.



Şekil 6.1: 12 elemanlı dizi şekilsel gösterimi

- Köşeli parantez içindeki konum numarası, ögenin dizini veya alt simgesi olarak adlandırılır. Dizin, bir tamsayı veya bir tamsayı ifadesi olmalıdır. Örneğin, ifade

`c[2] = 1000;`

`c[2]` dizi öğesine 1000 atar.

Benzer şekilde `a = 5` ve `b = 6` ise,

`c[a + b] += 2;`

bu durumda ifade, `c[11]` dizi öğesine 2 ekler.

- `c` dizisinin ilk üç öğesinin içeriği değerlerin toplamını yazdırmak için şunu yazarız:

```
printf("%d", c[0] + c[1] + c[2]);
```

- `c` dizisinin 6. öğesinin değerini 2'ye bölmek ve sonucu `x` değişkenine atamak için şunu yazarız:

```
x = c[6] / 2;
```

- Bir dizinin indeksini çevrelemek için kullanılan köşeli parantezler, aslında C'de bir operatör olarak kabul edilir. Bunlar, işlev (fonksiyon) çağrıma işlemciyle (operatörü ile) aynı önceliğe sahiptir. Şekil 2, operatörlerin önceliğini ve ilişkilendirilebilirliğini gösterir.

Operators	İlişkilendirilebilirlik	Tip
[] () ++ (postfix) -- (postfix)	soldan sağa	en yüksek
+ - ! ++ (prefix) -- (prefix) (type)	sağdan sola	tekli
* / %	soldan sağa	çarpımsal
+ -	soldan sağa	toplamsal
< <= > >=	soldan sağa	ilişkisel
== !=	soldan sağa	eşitlik
&&	soldan sağa	Mantıksal AND
	soldan sağa	Mantıksal OR
? :	sağdan sola	koşullu
= += -= *= /= %=	sağdan sola	atama
,	soldan sağa	virgül

Şekil 6.2: Operatör önceliği ve ilişkilendirilebilirlik (İşlem önceliğinin soldan sağa ya da sağdan sola olduğunu gösterir)

- Diziler bellekte yer kaplar. Bilgisayarın uygun miktarda bellek ayırmaması için her bir öğenin türünü ve her dizinin gerektirdiği öğe sayısını belirtirsiniz. Aşağıdaki tanım, 0-11 aralığında indekslere sahip olan c tamsayı dizisi için 12 eleman ayrıır.

```
int c[12];
```

- Aşağıdaki ifade, b tamsayı dizisi için 100 öğe ve x tamsayı dizisi için 27 öğe ayırır. Bu dizilerin sırasıyla 0-99 ve 0-26 aralığında indeksleri vardır. Aynı anda birden fazla dizi tanımlayabilmenize rağmen, her satırda yalnızca bir tane tanımlama tercih edilir, böylece her dizinin amacını açıklayan bir yorum ekleyebilirsiniz.

```
int b[100], x[27];
```

- Diziler başka veri türleri içerebilir. Örneğin, char türündeki bir dizi, bir karakter dizisini saklayabilir. Karakter dizileri ve dizilere benzerlikleri, işaretçiler ve diziler arasındaki ilişki ileriki bölümlerde tartışılacaktır.

Diziler ile ilgili bir örnek yapalım. Diğer tüm değişkenler gibi, başlatılmamış dizi öğeleri de çöp değerler içerir. Şekil 3, beş elemanlı bir tamsayı dizisi n'nin öğelerini sıfırlara ayarlamak (11-13. satırlar) ve diziyi tablo biçiminde (18-20. satırlar) yazdırmak için "for" ifadelerini kullanır. İlk "printf" ifadesi (satır 15), sonraki "for" ifadesinde yazdırılan iki sütun için sütun başlıklarını görüntüler.

```

1 // Fig. 6.3: fig06_03.c
2 // Initializing the elements of an array to zeros.
3 #include <stdio.h>
4
5 // function main begins program execution
6 int main(void)
7 {
8     int n[5]; // n is an array of five integers
9
10    // set elements of array n to 0
11    for (size_t i = 0; i < 5; ++i) {
12        n[i] = 0; // set element at location i to 0
13    }
14
15    printf("%s%13s\n", "Element", "Value");
16
17    // output contents of array n in tabular format
18    for (size_t i = 0; i < 5; ++i) {
19        printf("%7u%13d\n", i, n[i]);
20    }
21 }
```

Şekil 6.3: Bir dizinin öğelerini sıfırlama.

Şimdi bu örneği C de derleyip çalıştıralım ve “n[i]”, “i” değişkenini, dizi büyüğünü değiştirerek programın nasıl tepki verdiği ve sonucun nasıl değiştiğini görelim.

Sayaç-kontrol değişkeni i'nin her bir “for” ifadesinde (satır 11 ve 18) “size_t” türünde bildirildiğine dikkat edin, bu C standardına göre işaretsız bir veri türü temsil eder. Bu tür, bir dizinin boyutunu veya dizinlerini temsil eden tüm değişkenler için önerilir. “Size_t” tipi, genellikle diğer başlıklar (<stdio.h> gibi) tarafından dahil edilen <stddef.h> başlığında tanımlanır. [Not: Şekil 6.3'ü derlemeye çalışır ve hata alırsanız, programınıza <stddef.h> ekleyin.]

- Bir dizinin öğeleri, eşittir işareti ve virgülle ayrılmış bir dizi başlatıcı listesi içeren parantez {} ile tanımlandığında da başlatılabilir. Şekil 4, beş değerli (satır 9) bir tamsayı dizisini başlatır ve diziyi tablo biçiminde yazdırır.

```

1 // Fig. 6.4: fig06_04.c
2 // Initializing the elements of an array with an initializer list.
3 #include <stdio.h>
4
5 // function main begins program execution
6 int main(void)
7 {
8     // use initializer list to initialize array n
9     int n[5] = {32, 27, 64, 18, 95};
10
11    printf("%s%13s\n", "Element", "Value");
12
13    // output contents of array in tabular format
14    for (size_t i = 0; i < 5; ++i) {
15        printf("%7u%13d\n", i, n[i]);
16    }
17}

```

Element	Value
0	32
1	27
2	64
3	18
4	95

Şekil 6.4: Beş değerli bir tamsayı dizisini başlatır ve diziyi tablo biçiminde yazdırır.

Şimdi bu programı yazıp derleyerek sonuçları görelim. “n[5]” dizisinin değerini ve “i” değişkenini değiştirerek sonuçların nasıl değiştığını irdeleyelim.

- Dizideki öğelerden daha az başlatıcı varsa, kalan öğeler sıfır olarak başlatılır. Örneğin, Şekil 3'teki n dizisinin öğeleri, aşağıdaki gibi sıfır olarak başlatılabilir:

```
int n[10] = {0}; // initializes entire array to zeros
```

- Bu, ilk öğeyi sıfır olarak başlatır ve dizideki öğelerden daha az başlatıcı olduğundan, kalan dokuz öğeyi sıfır olarak başlatır. Diziler otomatik olarak sıfır olarak başlatılmaz. Kalan öğelerin otomatik olarak sıfırlanması için en azından ilk öğeyi sıfıra sıfırlamalısınız. Dizi öğeleri, statik diziler için program başlangıcından önce ve otomatik diziler için çalışma zamanında başlatılır.
- Dizi boyutu, başlatıcı listesine yazılmazsa, dizideki öğelerin sayısı, başlatıcı listesindeki öğelerin sayısı olacaktır. Örneğin, aşağıdaki ifade, belirtilen değerlerle başlatılan beş öğeli bir dizi oluşturur.

```
int n[] = {1, 2, 3, 4, 5};
```

- Bir Dizinin Boyutu Sembolik Bir Sabitle Belirtilebilir ve Hesaplamlarla Dizi Öğelerini Başlatma yapılabilir. Örneğin Şekil 5, beş öğeli bir dizinin öğelerini 2, 4, 6, ..., 10 değerlerine başlatır ve diziyi tablo biçiminde yazdırır. Değerler, döngü sayacı 2 ile çarpılıp 2 eklenecek oluşturulur.

```

1 // Fig. 6.5: fig06_05.c
2 // Initializing the elements of array s to the even integers from 2 to 10.
3 #include <stdio.h>
4 #define SIZE 5 // maximum size of array
5
6 // function main begins program execution
7 int main(void)
8 {
9     // symbolic constant SIZE can be used to specify array size
10    int s[SIZE]; // array s has SIZE elements
11
12    for (size_t j = 0; j < SIZE; ++j) { // set the values
13        s[j] = 2 + 2 * j;
14    }
15
16    printf("%s%13s\n", "Element", "Value");
17
18    // output contents of array s in tabular format
19    for (size_t j = 0; j < SIZE; ++j) {
20        printf("%7u%13d\n", j, s[j]);
21    }
22 }
```

Element	Value
0	2
1	4
2	6
3	8
4	10

Şekil 6.5: Bu örnekte “s” dizisinin öğeleri 2'den 10'a kadar çift tamsayılara başlatılıyor.

Şimdi bu programı yazıp derleyerek sonuçları görelim. “s[j]” dizisinin değerini ve “i” değişkenini, SIZE 5 dizi boyutunu değiştirerek sonuçların nasıl değiştığıni irdeleyelim.

- Bu programda #define önişlemci yönergesi tanıtılmıştır.

#define SIZE 5

- 4. satır, değeri 5 olan SIZE sembolik sabitini tanımlar. Sembolik sabit, program derlenmeden önce C önişlemci tarafından değiştirilen metinle değiştirilen bir tanımlayıcıdır. Program önceden işlendiğinde, SIZE sembolik sabitinin tüm oluşumları yerine geçen metin 5 ile değiştirilir. Dizi boyutlarını belirtmek için sembolik sabitlerin kullanılması, programları daha değiştirilebilir hale getirir.
- Şekil 5'te, #define direktifindeki SIZE değerini 5'ten 1000'e değiştirerek ilk “for” döngüsünü (satır 12) 1000 elemanlı bir diziyi doldurabilirdik. Eğer SIZE sembolik sabiti kullanılmamasıydı, 10, 12 ve 19. satırlardaki programı değiştirmek zorunda kalacaktık. Programlar büyükçe, bu teknik açık, okunması kolay, bakımı yapılabilir programlar yazmak için daha kullanışlı hale gelir. Sembolik bir sabitin (SIZE gibi) anlaşılması sayısalдан daha kolaydır.

Örnek Problem 1;

Bu örneğimizde, bir ankette toplanan verilerin sonuçlarını özetlemek için diziler kullanılıyor. Kırk öğrenciden öğrenci yemekhanesindeki yemeklerin kalitesini 1'den 10'a kadar (1 berbat ve 10 mükemmel anlamına gelir) derecelendirmeleri istenmiştir. 40 öğrencinin verdiği yanıtı bir tamsayı dizisine yerleştirin ve anketin sonuçlarını özetleyin. Yani hangi yanıt kaç defa verilmiş. (Örneğin kaç kişi 4 vermiş yada kaç kişi 7 vermiş gibi.)

Bu, tipik bir dizi uygulamasıdır (Şekil 7). 40 elemanlı yanıt dizisi (14-16. satırlar) öğrencilerin yanıtlarını içerir. Her yanıtın oluşum sayısını saymak için 11 öğeli bir “frequency” dizisi (satır 11) kullanıyoruz. frequency[0]'ı göz ardı ediyoruz, çünkü frequency[0] yerine yanıt 1'in frequency[1] de olması mantıklıdır. Bu, her yanıtı doğrudan frequency dizisindeki dizin olarak kullanmamıza izin verir.

```
1 // Fig. 6.7: fig06_07.c
2 // Analyzing a student poll.
3 #include <stdio.h>
4 #define RESPONSES_SIZE 40 // define array sizes
5 #define FREQUENCY_SIZE 11
6
7 // function main begins program execution
8 int main(void)
9 {
10    // initialize frequency counters to 0
11    int frequency[FREQUENCY_SIZE] = {0};
12
13    // place the survey responses in the responses array
14    int responses[RESPONSES_SIZE] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
15        1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
16        5, 6, 7, 5, 6, 4, 8, 6, 8, 10};
17
18    // for each answer, select value of an element of array responses
19    // and use that value as an index in array frequency to
20    // determine element to increment
21    for (size_t answer = 0; answer < RESPONSES_SIZE; ++answer) {
22        ++frequency[responses[answer]];
23    }
24
25    // display results
26    printf("%s%17s\n", "Rating", "Frequency");
27
28    // output the frequencies in a tabular format
29    for (size_t rating = 1; rating < FREQUENCY_SIZE; ++rating) {
30        printf("%6d%17d\n", rating, frequency[rating]);
31    }
32 }
```

Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

Şekil 6.7: Bir öğrenci anketini analiz etme.

Şimdi bu programı yazıp derleyerek sonuçları görelim. Programdaki öğrenci sayısını puanlama aralığını değiştirerek sonuçların nasıl değiştigini irdeleyelim.

Programda “for” döngüsü (satır 21–23), yanıtları responses dizi yanıtlarından birer birer alır ve “frequency” dizisindeki 10 sayıdan birini (frequency[1] ila frequency[10]) artırır. Döngüdeki anahtar ifade, ifade yanıtlarının değerine bağlı olarak uygun “frequency” sayacını artıran 22. Satırıdır.

```
++ frequency [responses [answer]];
```

Sayaç değişkeni “answer” 0 olduğunda, responses[answer] 1'dir, dolayısıyla

```
++ frequency[responses[answer]]; dizi ögesi 1'in bir artırılacağını gösterir.
```

```
++ frequency[1];
```

Yanıt 1 olduğunda, responses[answer] değeri

2, yani ++ frequency[responses[answer]]; dizi elemanı 2'nin artacağı şeklinde yorumlanır.

```
++ frequency [2];
```

answer 2 olduğunda, responses[answer] değeri 6'dır, dolayısıyla ++ frequency[responses[answer]]; dizi ögesi 6'nın artacağı şeklinde yorumlanır... vb.

```
++ frequency[6];
```

Ankette işlenen yanıtların sayısına bakılmaksızın, sonuçları özetlemek için yalnızca 11 ögeli bir dizi gerekir (sıfır ögesi yok sayılır). Veriler 13 gibi geçersiz değerler içeriyorsa, program frequency[13]. dizisine 1 eklemeye çalışır. Bu, dizinin sınırları dışında olacaktır. C'de, programın var olmayan bir ögeye atıfta bulunmasını önlemek için kontrol eden bir dizi sınırı yoktur.

Böylece, çalışan bir program bir dizinin her iki ucundan herhangi bir uyarı vermeden "çıkabilir" - bu, ilerdeki bölümlerde tartışacağımız bir güvenlik sorunudur. Tüm dizi referanslarının dizinin sınırları içinde kalmasını sağlamalısınız.

ÖDEV_1: Şekil 7 deki programı, öğrenci sayılarını, puan aralığını ve öğrencilerin yanıtlarını klavyeden girilecek şekilde yeniden düzenleyin. Bana mail ile gönderin (celaleddin.yeroglu@iste.edu.tr)

Programı bu şekilde düzenleyen ve programı doğru çalışan ilk 10 kişiye vize sınavında fazladan “5” puan eklenecektir.