Karakter Dizileri

Şimdiye kadar yalnızca tamsayı dizilerini tartıştık. Fakat diziler her türden veriyi tutabilir. Şimdi dizileri karakter dizilerinde depolamayı tartışalım. Şimdiye kadar sahip olduğumuz tek dizi işleme yeteneğimiz, printf ile bir dizi çıktısı almaktır. "Merhaba" gibi bir dize, gerçekte C'deki tek tek karakterlerden oluşan bir dizidir.

Karakter dizilerinin birkaç özelliği vardır. Bir karakter dizisi, bir dize sabit değeri kullanılarak başlatılabilir. Örneğin, aşağıdaki ifade, “string1” dizisinin öğelerini, "first" dize sabitlerindeki tek tek karakterler ile başlatır.

char string1[] = "first";

Bu durumda, “string1” dizisinin boyutu, dizenin uzunluğuna göre derleyici tarafından belirlenir. "First" dizesi, beş karakter artı boş karakter adı verilen özel bir dize sonlandırma karakteri içerir. Böylece, “string1” dizisi aslında altı öğe içerir.

Boş karakteri temsil eden '\0'dır. C'deki tüm diziler bu karakterle biter. Bir dizeyi temsil eden bir karakter dizisi her zaman dizedeki karakter sayısını ve sonlandırıcı boş karakteri tutacak kadar büyük tanımlanmalıdır.

char string1[] = {'f', 'i', 'r', 's', 't', '\0'};

Bir dize gerçekten bir karakter dizisi olduğundan, bir dizedeki tek tek karakterlere doğrudan dizi gösterimini kullanarak erişebiliriz. Örneğin, string1[0], 'f' karakteridir ve string1[3], 's' karakteridir.

Ayrıca "scanf" ve %s belirleyicisini kullanarak klavyeden bir karakter dizisine doğrudan bir dize girebiliriz. Örneğin,

char string2[20];

en fazla 19 karakterlik bir diziyi ve sonlandırıcı boş karakteri depolayabilen bir karakter dizisi oluşturur.

scanf("%19s", string2);

İfadesi, klavyeden string2'ye bir “string” okur. &, normalde scanf'e bir değişkenin bellekteki konumunu sağlamak için kullanılır, böylece bir değer orada saklanabilir. “scanf” işlevi, bir boşluk, sekme, yeni satır veya dosya sonu göstergesiyle karşılaşılıncaya kadar karakterleri okuyacaktır. Sonlandırıcı boş karaktere yer bırakmak için “string2” dizisi 19 karakterden uzun olmamalıdır. Kullanıcı 20 veya daha fazla karakter yazarsa programınız çökebilir veya arabellek taşması adı verilen bir güvenlik açığı oluşturabilir. Bu nedenle, %19s belirleyicisini kullanılabilir, böylece scanf en fazla 19 karakter okuyabilir ve string2 dizisinin sonundan sonra belleğe karakter yazmaz.

“scanf” fonksiyonu, dizinin ne kadar büyük olduğunu kontrol etmez. Böylece “scanf”, dizinin sonunun ötesine yazabilir.

Bir dizeyi temsil eden bir karakter dizisi, printf ve %s belirleyicisi ile yazdırılabilir. Örneğin “string2” dizisi şu ifadeyle yazdırılır

printf("%s\n", string2);

“printf” fonksiyonu, “scanf” gibi, karakter dizisinin ne kadar büyük olduğunu kontrol etmez. Dizenin karakterleri, sonlandırıcı bir boş karakterle karşılaşılıncaya kadar yazdırılır.

Şekil 6.10, bir karakter dizisini bir sabit dize ile başlatmayı, bir karakter dizisine bir dize okumayı, bir karakter dizisini bir dize olarak yazdırmayı ve bir dizenin tek tek karakterlerine erişmeyi gösterir. Program, “string1” dizisinde döngü yapmak ve %c belirticisini kullanarak boşluklarla ayrılmış tek tek karakterleri yazdırmak için bir “for” ifadesi (22-24. satırlar) kullanır.

```
1  // Fig. 6.10: fig06_10.c
2  // Treating character arrays as strings.
3  #include <stdio.h>
4  #define SIZE 20
5
6  // function main begins program execution
7  int main(void)
8  {
9  char string1[SIZE]; // reserves 20 characters
10 char string2[] = "string literal"; // reserves 15 characters
11
12 // read string from user into array string1
13 printf("%s", "Enter a string (no longer than 19 characters): ");
14 scanf("%19s", string1); // input no more than 19 characters
15
16 // output strings
17 printf("string1 is: %s\nstring2 is: %s\n"
18  "string1 with spaces between characters is:\n",
19  string1, string2);
20
21 // output characters until null character is reached
22     for (size_t i = 0; i < SIZE && string1[i] != '\0'; ++i) {
23         printf("%c ", string1[i]);
24         }
25     puts("");
26 }
```

Şekil 6.10 Karakter dizilerini " strings" olarak ele alma.

Şimdi bu programı yazıp derleyelim ve çalıştıralım. Karakterler arasına boşluk koyduğumuzda ne oluyor bakalım. Eğer boşluk bırakırsak boşluktan sonraki karakterleri yazdırmadığına dikkat edelim. Burada karakter dizilerinde Türkçe karakterler kullansak ta bu karakterlerin hata vermeden yazdırıldığına dikkat edelim.

Merhaba ben bilgisayar mühendisiyim

Statik Yerel Diziler ve Otomatik Yerel Diziler

Programın süresi boyunca statik bir yerel değişken bulunur, ancak yalnızca fonksiyon içerisinde görünür. Yerel bir dizi tanımlarken statik dizi oluşturabiliriz, böylece fonksiyon her çağrıldığında dizi yeniden oluşturulmaz ve yeniden başlatılmaz ve programda fonksiyondan her çıkıldığında dizi yok edilmez Dizi ilk halini korur. Bu, özellikle büyük diziler içeren sık çağrılan fonksiyonlara sahip programlar için program yürütme süresini azaltır.

Statik olan diziler, program başlangıcında bir kez başlatılır. O dizinin öğeleri varsayılan olarak sıfır olarak başlatılır.

Şekil 6.11, yerel statik dizili (satır 24) staticArrayInit (satır 21–39) fonksiyonunu ve yerel otomatik dizili (satır 45) automaticArrayInit (satır 42–60) fonksiyonunu göstermektedir. Şekil 6.11'de verilen progrmda "staticArrayInit" fonksiyonu iki kez çağrılır (satır 12 ve 16). Fonksiyondaki yerel statik dizi, program başlangıcından önce sıfır olarak başlatılır (satır 24).

Fonksiyon diziyi yazdırır, her öğeye 5 ekler ve diziyi yeniden yazdırır. Fonksiyon ikinci kez çağrıldığında, statik dizi, ilk fonksiyon çağrısı sırasında saklanan değerleri içerir.

"automaticArrayInit" fonksiyonu ayrıca iki kez çağrılır (satır 13 ve 17). Fonksiyondaki otomatik yerel dizinin öğeleri 1, 2 ve 3 değerleri ile başlatılır (satır 45).

Fonksiyon diziyi yazdırır, her öğeye 5 ekler ve diziyi yeniden yazdırır. Fonksiyon ikinci kez çağrıldığında, dizinin otomatik depolama süresi olduğundan dizi öğeleri yeniden 1, 2 ve 3 olarak başlatılır.

Şimdi programı bölüm bölüm inceleyelim. İlk Bölümde programda fonksiyonların tanımlanması ve ana programdan (main) fonksiyonların çağrılması yapıldığına dikkat edelim.

```
1  // Fig. 6.11: fig06_11.c
2  // Static arrays are initialized to zero if not explicitly initialized.
3  #include <stdio.h>
4
5  void staticArrayInit(void); // function prototype
6  void automaticArrayInit(void); // function prototype
7
8  // function main begins program execution
9  int main(void)
10 {
11    puts("First call to each function:");
12    staticArrayInit();
13    automaticArrayInit();
14
15    puts("\n\nSecond call to each function:");
16    staticArrayInit();
17    automaticArrayInit();
18 }
```

Bu ana programda istediğimiz fonksiyonu istediğimiz kadar çağırabiliriz.

İkinci bölümde ise statik dizi fonksiyonunun (staticArrayInit) çalışması ve “array1” dizisinde yapılan işlemler verilmektedir.

```
19
20  // function to demonstrate a static local array
21  void staticArrayInit(void)
22  {
23     // initializes elements to 0 before the function is called
24     static int array1[3];
25
26     puts("\nValues on entering staticArrayInit:");
27
28     // output contents of array1
29     for (size_t i = 0; i <= 2; ++i) {
30        printf("array1[%u] = %d  ", i, array1[i]);
31     }
32
33     puts("\nValues on exiting staticArrayInit:");
34
35     // modify and output contents of array1
36     for (size_t i = 0; i <= 2; ++i) {
37        printf("array1[%u] = %d  ", i, array1[i] += 5);
38     }
39  }
```

Üçüncü bölümde ise otomatik dizi fonksiyonunun (automaticArrayInit) çalışması ve “array2” dizisinde yapılan işlemler gösterilmektedir.

```
40
41  // function to demonstrate an automatic local array
42  void automaticArrayInit(void)
43  {
44     // initializes elements each time function is called
45     int array2[3] = {1, 2, 3};
46
47     puts("\n\nValues on entering automaticArrayInit:");
48
49     // output contents of array2
50     for (size_t i = 0; i <= 2; ++i) {
51        printf("array2[%u] = %d  ", i, array2[i]);
52     }
53
```

Bu örnekte görüldüğü gibi ana program içerisinde istediğimiz kadar fonksiyon tanımlayıp bu fonksiyonları istediğimiz kadar çağırabiliriz. Bu örnekte iki farklı dizi için iki fonksiyon oluşturulmuştur. Program çalıştığında görüleceği gibi program her çağrıldığında otomatik dizi “array2” dizisinin her defasında ilk yüklenen değerleri tekrar aldığına dikkat edelim. Yani fonksiyon her çağrıldığında dizi orijinal değerlerine dönmektedir.

Statik dizi “array1” de ise durum farklıdır. Dizi en son hesaplanan değeri tutar ve yeni yapılan işlemler en son değer üzerinde yapılır.

Şimdi bu programı yeniden düzenleyerek tekrar oluşturalım. Örneğin statik diziyi 5 defa otomatik diziyi 3 defa çağıralım ve sonuçların nasıl değiştiğini görelim.

Bölüm 1 de fonksiyon tanımlama ve ana programdan fonksiyonların çağrılması:

```c
// Fig. 6.11: fig06_11.c
// Eğer özel olarak belirtilmediyse Statik diziler 0 olarak başlatılır.
#include <stdio.h>

void staticArrayInit(void); // function prototype
void automaticArrayInit(void); // function prototype

// function main begins program execution
int main(void)
{
    puts("Her fonksiyonun ilk cagrilisi:");
    staticArrayInit();
    automaticArrayInit();

    puts("\n\n Her fonksiyonun ikinci cagrilisi:");
    staticArrayInit();
    automaticArrayInit();

    puts("\n\n Her fonksiyonun ucuncu cagrilisi:");
    staticArrayInit();
    automaticArrayInit();

    puts("\n\n Statik dizi fonksiyonunun dorduncu ve besinci cagrilisi:");
    staticArrayInit();
    staticArrayInit();
}
```

Bölüm 2: statik dizi fonksiyonunun (staticArrayInit) çalışması ve “array1” dizisinde yapılan işlemler.

```c

// Statik yerel diziyi gösteren fonksiyon
//(function to demonstrate a static local array)
void staticArrayInit(void)
{
    // Fonksiyon çagrilmadan once elemanlarini 0 ile başlatır. 3 birimlik yer ayirir.
    //(initializes elements to 0 before the function is called)
    static int array1[3];

    puts("\n staticArrayInit fonksiyonuna deger girilmesi:");

    // dizi1 içeriğinin yazdırılması (output contents of array1)
    for (size_t i = 0; i <= 2; ++i) {
        printf("array1[%u] = %d ", i, array1[i]);
    }

    puts("\n staticArrayInit fonksiyonunda olan değerler:");

    // dizi1 içeriğinin düzenlenmesi ve ciktisi (modify and output contents of array1)
    for (size_t i = 0; i <= 2; ++i) {
        printf("array1[%u] = %d ", i, array1[i] += 5);
    }
}
```

Bölüm 3: otomatik dizi fonksiyonunun (automaticArrayInit) çalışması ve “array2” dizisinde yapılan işlemler.

```
50
51  // Otomatiik yerel dizinin gösterimi (function to demonstrate an automatic local array)
52  void automaticArrayInit(void)
53  {
54      // Fonksiyon her çağrıldığında bu değerlerle başlatır
55      //(initializes elements each time function is called)
56      int array2[3] = {1, 2, 3};
57
58      puts("\n\n automaticArrayInit fonksiyonuna girilen degerler:");
59
60      // dizi2 nin içeriği (output contents of array2)
61      for (size_t i = 0; i <= 2; ++i) {
62          printf("array2[%u] = %d ", i, array2[i]);
63  }
64
65      puts("\n automaticArrayInit fonksiyonundaki degerler:");
66
67      // dizi2 içeriğinin düzenlenmesi ve çıktısı
68      //(modify and output contents of array2)
69      for (size_t i = 0; i <= 2; ++i) {
70          printf("array2[%u] = %d ", i, array2[i] += 5);
71      }
72  }
```

```
Her fonksiyonun ilk cagrilisi:

 staticArrayInit fonksiyonuna deger girilmesi:
array1[0] = 0 array1[1] = 0 array1[2] = 0
 staticArrayInit fonksiyonunda olan de-erler:
array1[0] = 5 array1[1] = 5 array1[2] = 5

 automaticArrayInit fonksiyonuna girilen degerler:
array2[0] = 1 array2[1] = 2 array2[2] = 3
 automaticArrayInit fonksiyonundaki degerler:
array2[0] = 6 array2[1] = 7 array2[2] = 8

 Her fonksiyonun ikinci cagrilisi:

 staticArrayInit fonksiyonuna deger girilmesi:
array1[0] = 5 array1[1] = 5 array1[2] = 5
 staticArrayInit fonksiyonunda olan de-erler:
array1[0] = 10 array1[1] = 10 array1[2] = 10

 automaticArrayInit fonksiyonuna girilen degerler:
array2[0] = 1 array2[1] = 2 array2[2] = 3
 automaticArrayInit fonksiyonundaki degerler:
array2[0] = 6 array2[1] = 7 array2[2] = 8

 Her fonksiyonun ucuncu cagrilisi:

 staticArrayInit fonksiyonuna deger girilmesi:
array1[0] = 10 array1[1] = 10 array1[2] = 10
 staticArrayInit fonksiyonunda olan de-erler:
array1[0] = 15 array1[1] = 15 array1[2] = 15

 automaticArrayInit fonksiyonuna girilen degerler:
array2[0] = 1 array2[1] = 2 array2[2] = 3
 automaticArrayInit fonksiyonundaki degerler:
array2[0] = 6 array2[1] = 7 array2[2] = 8

 Statik dizi fonksiyonunun dorduncu ve besinci cagrilisi:

 staticArrayInit fonksiyonuna deger girilmesi:
array1[0] = 15 array1[1] = 15 array1[2] = 15
 staticArrayInit fonksiyonunda olan de-erler:
array1[0] = 20 array1[1] = 20 array1[2] = 20
 staticArrayInit fonksiyonuna deger girilmesi:
array1[0] = 20 array1[1] = 20 array1[2] = 20
 staticArrayInit fonksiyonunda olan de-erler:
array1[0] = 25 array1[1] = 25 array1[2] = 25
```

Statik dizi 4ncü çağrılışta değeri 15 idi işlemden sonra 20 oldu. 5nci çağrılışta ise içindeki değer 20 idi işlemden sonra 25 oldu.