BLM-111 PROGRAMLAMA DİLLERİ I

Ders-12 Fonksiyonlar ve Kapsama Kuralları

Yrd. Doç. Dr. Ümit ATİLA
umitatila@karabuk.edu.tr
http://web.karabuk.edu.tr/umitatilla/

Fonksiyonlar

- Fonksiyonlar
  - C 'de modüller
  - Programlar kullanıcı tanımlı fonksiyonları ve kütüphane fonksiyonlarını birlikte kullanırlar.
  - C standart kütüphanesi zengin bir fonksiyon çeşitliliğine sahiptir.

Fonksiyonların Faydaları

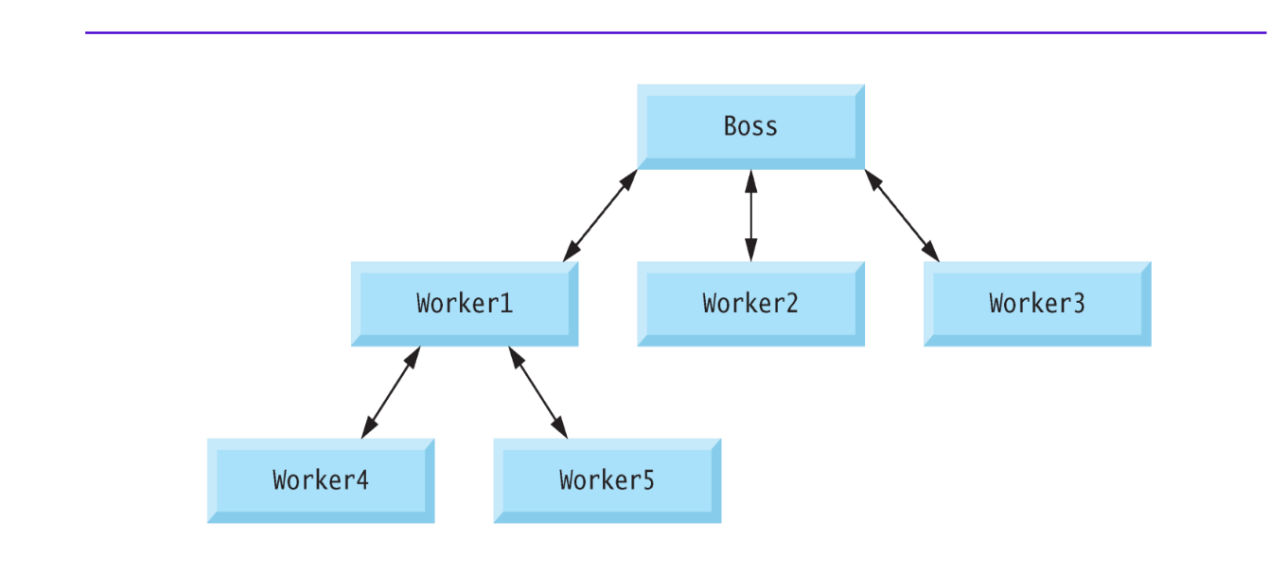
- Fonksiyonları faydaları
  - Böl ve yönet
    - Küçük parçalar veya bileşenlerden program oluştur.
    - Bu küçük parçalara modül denir.
    - Fonksiyonlar bir programı modüler hale getirir.
    - Her bir modül programın kendisinden daha yönetilebilirdir.
  - Yazılımın tekrar kullanılabilirliği
    - Mevcut fonksiyonlarınızı yeni bir program yapımında kullanabilirsiniz
    - Soyutlama – iç detaylar gizlenir(kütüphane fonksiyonları)
  - Kod tekrarı önlenir.

Fonksiyonlar

- Fonksiyonlar içinde tanımlanan tüm değişkenler yerel değişkenlerdir
  - Sadece tanımlandıkları fonksiyon içinde geçerlidirler
- Parametreler
  - Fonksiyonlar ile haberleşmede kullanılan bilgi
  - Yerel değişkendirler
- Fonksiyon çağırılması
  - Fonksiyon ismini ve argümanlarını (veri) belirt.
  - Fonksiyonlar birtakım işlemler gerçekleştirir.
  - Fonksiyonlar sonuç döndürürler

Fonksiyonlar

- Fonksiyon çağırma analojisi
  - Patron işçiden bir işi yapmasını ister
  - İşçi bilgi toplar, işi yapar ve sonucu patrona bildirir.
  - Bilgi gizleme: patron işin detaylarını bilmez.



Fonksiyon Tanımlama

- Fonksiyon tanımlama formatı:

```
Geri_dönüş_değer_tipi   fonksiyon _adı    ( parametre-liste )
{
   tanımlamalar ve ifadeler
}
```

- *Fonksiyon-adı* herhangi bir geçerli tanımlayıcı olabilir
- *Geri dönüş değer tipi* fonksiyonu çağırana döndürülen sonucun veri tipidir.
- *Geri dönüş değer tipi* void ise fonksiyon herhangi bir değer döndürmez.
- *Geri dönüş değer tipi, fonksiyon_adı* ve *parametre-liste* üçlüsü fonksiyon başlığı (header)

Fonksiyon Tanımlama

- *parametre-liste* fonksiyonun çağırılırken aldığı parametreleri tanımlayan, virgülle birbirinden ayrılmış bir listedir.
- Eğer fonksiyon herhangi bir parametre almıyorsa, parametre listesi void olur.
- Her bir parametrenin tipi belirtilmelidir.

Fonksiyon Tanımlama

- Kırlangıç parantezler arasındaki *Tanımlamalar* ve *ifadeler* fonksiyon gövdesi.
- Fonksiyon gövdesinin diğer bir adı blok.
- Değişkenler herhangi bir blok içerisinde tanımlanabilir ve bloklar iç içe olabilir.
- Bir fonksiyon diğer bir fonksiyonun içerisinde tanımlanamaz.

Fonksiyon Tanımlama

- Çağırılan bir fonksiyondan, fonksiyonun çağırıldığı noktaya kontrolün iade edilmesinin üç yöntemi vardır.
- Eğer fonksiyon herhangi bir sonuç döndürmüyorsa,
  - Sağ kırlangıç parantez ile kontrol iade edilmiş olunur.
  - Ya da basitçe return; ifadesi çalıştırılır.
- Eğer fonksiyon bir sonuç döndürüyorsa,
  - return *ifade;*
  - İfadenin değerini fonksiyonu çağırana döndürür.

Fonksiyon Tanımlama

```c
/* Fig. 5.4: fig05_04.c
   Finding the maximum of three integers */
#include <stdio.h>

int maximum( int x, int y, int z ); /* function prototype */

/* function main begins program execution */
int main( void )
{
   int number1; /* first integer */
   int number2; /* second integer */
   int number3; /* third integer */

   printf( "Enter three integers: " );
   scanf( "%d%d%d", &number1, &number2, &number3 );

   /* number1, number2 and number3 are arguments
      to the maximum function call */
   printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );
   return 0; /* indicates successful termination */
} /* end main */

/* Function maximum definition */
/* x, y and z are parameters */
int maximum( int x, int y, int z )
{
   int max = x; /* assume x is largest */

   if ( y > max ) { /* if y is larger than max, assign y to max */
      max = y;
   } /* end if */

   if ( z > max ) { /* if z is larger than max, assign z to max */
      max = z;
   } /* end if */

   return max; /* max is largest value */
} /* end function maximum */
```

Fig. 5.4 | Finding the maximum of three integers. (Part 3 of 4.)

Fonksiyon Prototipi

- Bir fonksiyonun künyesidir.
- Eğer fonksiyon tanımlaması çağırımdan sonra ise prototip tanımlanmalıdır.
- Aşağıdaki prototipe sahip fonksiyon
  - `int maximum( int x, int y, int z );`
  - 3 tamsayı parametre alır.
  - Geriye tamsayı döndürür.

Fonksiyon Prototipi

- Bir fonksiyon çağırımı prototipi ile uyuşmuyorsa derleme hatası oluşur.
- Eğer fonksiyon prototipi ile fonksiyon tanımlaması uyuşmuyorsa da hata oluşur.
- Fonksiyon prototiplerinin diğer bir önemli özelliği ise argümanların uygun bir veri tipine zorlanmasıdır.
- Örneğin, matematik fonksiyonu sqrt <math.h> içerisinde yer alan prototipinde double belirtilmiş olsa da integer argümanla da çağırılabilir. Fonksiyon yine de doğru çalışacaktır
  - `printf( "%.3f\n", sqrt( 4 ) );`
  - İfadesindeki sqrt(4) doğru bir şekilde değerlendirilir ve 2.000 değeri yazdırılır.

Fonksiyon Parametre Terfi Kuralları

- Genelde, fonksiyon prototipindeki parametre tiplerine tam uymayan argüman değerleri fonksiyon çağırılmadan önce uygun tiplere dönüştürülürler.
- Eğer C'nin terfi kuralları takip edilemez ise bu dönüşümler yanlış sonuçlar doğurabilir.
- Terfi kuralları veri kaybı yaşamadan bir veri tipinden diğerine dönüşüm kurallarını tanımlar.

Fonksiyon Parametre Terfi Kuralları

Data type	printf conversion specification	scanf conversion specification
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
unsigned short	%hu	%hu
short	%hd	%hd
char	%c	%c

Başlık Header Dosyaları

- Kütüphane fonksiyonlarının prototiplerini barındırırlar.
- <stdlib.h> , <math.h> , vs
- #include <dosya_adı> ile yüklenir.
  - #include <math.h>
- Özel başlık dosyaları
  - Fonksiyonlar içeren bir dosya oluştur.
  - dosya_adı.h şeklinde isim ile kaydet.
  - Başka dosyalar içerisinde #include “dosya_adı.h“ olarak yükle.
  - Fonksiyonları tekrar kullan.

Başlık Header Dosyaları

- **math.h** → Matematik kütüphanesi
- **ctype.h** → Karakter özellikleri, küçük büyük harfe çevirme vs.
- **stdio.h** →Standart giriş / çıkış fonksiyonları
- **stdlib.h** → Sayıyı metnei metni sayıya dönüştürme, hafıza yönetimi, rasgele sayılar ve bazı diğer faydalı fonksiyonlar.
- **string.h** → String işlemleri
- **time.h** → Zaman ve tarih fonksiyonları

Matematik Fonksiyonları

- Matematik kütüphane fonksiyonları
  - Temel matematik hesaplamaları yapar.
  - `#include <math.h>`
- Fonksiyonları çağırmak için kullanılacak format
  - `FonksiyonAdı( argümanı );`
- Eğer birden fazla argüman varsa, aralarında virgül kullan
- Tüm matematik fonksiyonlar double veri tipi döndürür
- Argümanlar sabit, değişken veya ifade olabilir

Matematik Fonksiyonları

Function	Description	Example
`sqrt( x )`	square root of *x*	`sqrt( 900.0 )` is `30.0` `sqrt( 9.0 )` is `3.0`
`exp( x )`	exponential function e^x	`exp( 1.0 )` is `2.718282` `exp( 2.0 )` is `7.389056`
`log( x )`	natural logarithm of *x* (base *e*)	`log( 2.718282 )` is `1.0` `log( 7.389056 )` is `2.0`

`log10( x )`	logarithm of *x* (base 10)	`log10( 1.0 )` is `0.0`
		`log10( 10.0 )` is `1.0`
		`log10( 100.0 )` is `2.0`

`fabs( x )`	absolute value of *x*	`fabs( 13.5 )` is `13.5`
		`fabs( 0.0 )` is `0.0`
		`fabs( -13.5 )` is `13.5`

`ceil( x )`	rounds *x* to the smallest integer not less than *x*	`ceil( 9.2 )` is `10.0` `ceil( -9.8 )` is `-9.0`
`floor( x )`	rounds *x* to the largest integer not greater than *x*	`floor( 9.2 )` is `9.0` `floor( -9.8 )` is `-10.0`

Matematik Fonksiyonları

Function	Description	Example
`pow( x, y )`	x raised to power y (x^y)	`pow( 2, 7 )` is `128.0` `pow( 9, .5 )` is `3.0`
`fmod( x, y )`	remainder of x/y as a floating-point number	`fmod( 13.657, 2.333 )` is `1.992`
`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 )` is `0.0`
`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 )` is `1.0`
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 )` is `0.0`

Örnek: Kare alan fonksiyon

```c
#include <stdio.h>
float kareAl(float);

void main()
{
    int sayac;
    for(sayac = 1; sayac<=10; sayac++)
    {
        printf("Sayi:%d  Karesi:%d\n", sayac, kareAl(sayac));
    }

    printf("\n%.2f", kareAl(4.5));
}

float kareAl(float a)
{
    return a*a;
}
```

Örnek: Dört İşlem

```c
#include <stdio.h>
int toplam(int, int);
int cikar(int, int);
int carp(int, int);
float bol(int, int);

void main()
{
    int secim,s1,s2;
    while(1)
    {
        printf("1-Topla\n2-Cikar\n3-Carp\n4-Bol\n5-Cikis\n");
        scanf("%d", &secim);
        printf("Sayilari gir:");
        scanf("%d %d", &s1, &s2);

        if(secim == 1)
            printf("Sonuc = %d", topla(s1,s2));
        else if(secim == 2)
            printf("Sonuc = %d", cikar(s1,s2));
        else if(secim == 3)
            printf("Sonuc = %d", carp(s1,s2));
        else if(secim == 4)
            printf("Sonuc = %.2f", bol(s1,s2));
        else if(secim == 5)
            exit(0);
        else printf("Yanlis giris");
    }
}
```

```c
int topla(int a, int b)
{
    return a+b;
}
int cikar(int a, int b)
{
    return a-b;
}
int carp(int a, int b)
{
    return a*b;
}
float bol(int a, int b)
{
    return (float)a/b;
}
```

Örnek: Üs alma

```c
#include <stdio.h>
double usAl(double, double);

void main()
{
    double a,b;
    printf("Taban ve us degeri gir:");
    scanf("%lf %lf", &a, &b);
    printf("%.2f", usAl(a,b));
}

double usAl(double x, double y)
{
    int sayac;
    double sonuc=1.0;
    for(sayac=0;sayac<y;sayac++)
    {
        sonuc *= x;
    }
    return sonuc;
}
```

Saklama Sınıfları

- Nesne kendi bloğu içinde oluşturulur ve yok edilir
  - **auto**: yerel değişkenler için ön tanımlıdır
    - auto double x, y;
  - **register**: değişkeni yüksek hızlı kaydedicilere yerleştirmeye çalışır
    - register int counter= 1;

Saklama Sınıfları

- **Statik Saklama**
  - Değişken tüm programın çalışması sürecinde var olur.
  - Ön tanımlı değeri sıfırdır.
  - **static**: fonksiyonlar içinde tanımlanan yerel değişkenlerdir.
    - Fonksiyon sonlandıktan sonra değişken değeri saklanır
    - Sadece tanımlandıkları fonksiyonda geçerlidirler

Saklama Sınıfları

- **Dosya kapsama alanı**
  - Bir fonksiyon dışında tanımlanan bir tanımlayıcı dosya kapsama alanına sahiptir.
  - Böyle bir tanımlayıcı tanımlandıkları noktadan itibaren dosya sonuna kadar tüm fonksiyonlar içinde geçerlidir
  - Global değişkenler, fonksiyon tanımlamaları hep dosya kapsama alanına sahiptir.

Saklama Sınıfları

- **Blok Alanı**
  - Blok içinde tanımlanan tanımlayıcılar
  - Blok alanı tanımlandığı noktadan başlar sağ kırlangıç paranteze kadar devam eder.
  - Değişkenler, yerel değişkenler ve fonksiyonlar için kullanılır.
  - Dış bloklar eğer iç blokta aynı isimle değişken varsa iç bloktan gizlenirler.

Saklama Sınıfları

```c
/* Fig. 5.12: fig05_12.c
   A scoping example */
#include <stdio.h>

void useLocal( void ); /* function prototype */
void useStaticLocal( void ); /* function prototype */
void useGlobal( void ); /* function prototype */

int x = 1; /* global variable */

/* function main begins program execution */
int main( void )
{
   int x = 5; /* local variable to main */

   printf("local x in outer scope of main is %d\n", x );

   { /* start new scope */
      int x = 7; /* local variable to new scope */

      printf( "local x in inner scope of main is %d\n", x );
   } /* end new scope */

   printf( "local x in outer scope of main is %d\n", x );

   useLocal(); /* useLocal has automatic local x */
   useStaticLocal(); /* useStaticLocal has static local x */
   useGlobal(); /* useGlobal uses global x */
   useLocal(); /* useLocal reinitializes automatic local x */
   useStaticLocal(); /* static local x retains its prior value */
   useGlobal(); /* global x also retains its value */

   printf( "\nlocal x in main is %d\n", x );
   return 0; /* indicates successful termination */
} /* end main */

/* useLocal reinitializes local variable x during each call */
void useLocal( void )
{
   int x = 25; /* initialized each time useLocal is called */

   printf( "\nlocal x in useLocal is %d after entering useLocal\n", x );
   x++;
   printf( "local x in useLocal is %d before exiting useLocal\n", x );
} /* end function useLocal */
```

Saklama Sınıfları

```
47  /* useStaticLocal initializes static local variable x only the first time
48     the function is called; value of x is saved between calls to this
49     function */
50  void useStaticLocal( void )
51  {
52     /* initialized only first time useStaticLocal is called */
53     static int x = 50;
54
55     printf( "\nlocal static x is %d on entering useStaticLocal\n", x );
56     x++;
57     printf( "local static x is %d on exiting useStaticLocal\n", x );
58  } /* end function useStaticLocal */
59
60  /* function useGlobal modifies global variable x during each call */
61  void useGlobal( void )
62  {
63     printf( "\nglobal x is %d on entering useGlobal\n", x );
64     x *= 10;
65     printf( "global x is %d on exiting useGlobal\n", x );
66  } /* end function useGlobal */
```

```
local x in outer scope of main is 5
local x in inner scope of main is 7
local x in outer scope of main is 5

local x in useLocal is 25 after entering useLocal
local x in useLocal is 26 before exiting useLocal

local static x is 50 on entering useStaticLocal
local static x is 51 on exiting useStaticLocal

global x is 1 on entering useGlobal
global x is 10 on exiting useGlobal

local x in useLocal is 25 after entering useLocal
local x in useLocal is 26 before exiting useLocal

local static x is 51 on entering useStaticLocal
local static x is 52 on exiting useStaticLocal

global x is 10 on entering useGlobal
global x is 100 on exiting useGlobal

local x in main is 5
```

Dizilerin Fonksiyonlara Gönderilmesi

- Bir diziyi bir fonksiyona parametre olarak göndermek için parantez kullanmadan sadece dizinin ismi belirtilir.
  - **int** myArray [**24**];
  - **myFunction** (myArray, 24);
- Char dizilerinin aksine diğer türdeki diziler her hangi bir sonlandırma karakteri içermezler.
- Bu sebeple fonksiyonlara dizideki eleman sayısıda parametre olarak gönderilir ki, fonksiyon uygun sayıda eleman üzerinde işlem yapsın.

Dizilerin Fonksiyonlara Gönderilmesi

- Dizilerin fonksiyonlara gönderilmesi referans ile çağırma işlemidir (call by reference).
- Dizinin adı aslında ilk elemanının adresidir.
- Fonksiyon böylece dizinin ilk elemanının hafıza nerede olduğunu bilir.
  - Orijinal hafıza bölgesinde işlem yapılır.
- Dizideki her hangi bir elemanın fonksiyona gönderilmesi ise değer ile çağırmadır (call by value).
  - Fonksiyona herhangi bir indisteki elemanın değeri gönderilir
    - myArray [3]
- Bir int dizi ve bir int değeri parametre olarak alan fonksiyon prototipi;
  - void myArray (int [ ], int)

Dizilerin Fonksiyonlara Gönderilmesi

```
1  /* Fig. 6.13: fig06_13.c
2     Passing arrays and individual array elements to functions */
3  #include <stdio.h>
4  #define SIZE 5
5
6  /* function prototypes */
7  void modifyArray( int b[], int size );
8  void modifyElement( int e );
9
10 /* function main begins program execution */
11 int main( void )
12 {
13    int a[ SIZE ] = { 0, 1, 2, 3, 4 }; /* initialize a */
14    int i; /* counter */
15
16    printf( "Effects of passing entire array by reference:\n\nThe "
17       "values of the original array are:\n" );
18
19    /* output original array */
20    for ( i = 0; i < SIZE; i++ ) {
21       printf( "%3d", a[ i ] );
22    } /* end for */
23
```

Dizilerin Fonksiyonlara Gönderilmesi

```
24      printf( "\n" );
25
26      /* pass array a to modifyArray by reference */
27      modifyArray( a, SIZE );
28
29      printf( "The values of the modified array are:\n" );
30
31      /* output modified array */
32      for ( i = 0; i < SIZE; i++ ) {
33         printf( "%3d", a[ i ] );
34      } /* end for */
35
36      /* output value of a[ 3 ] */
37      printf( "\n\n\nEffects of passing array element "
38         "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
39
40      modifyElement( a[ 3 ] ); /* pass array element a[ 3 ] by value */
41
42      /* output value of a[ 3 ] */
43      printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
44      return 0; /* indicates successful termination */
45   } /* end main */
46
```

Dizilerin Fonksiyonlara Gönderilmesi

```
47  /* in function modifyArray, "b" points to the original array "a"
48     in memory */
49  void modifyArray( int b[], int size )
50  {
51     int j; /* counter */
52
53     /* multiply each array element by 2 */
54     for ( j = 0; j < size; j++ ) {
55        b[ j ] *= 2;
56     } /* end for */
57  } /* end function modifyArray */
58
59  /* in function modifyElement, "e" is a local copy of array element
60     a[ 3 ] passed from main */
61  void modifyElement( int e )
62  {
63     /* multiply parameter by 2 */
64     printf( "Value in modifyElement is %d\n", e *= 2 );
65  } /* end function modifyElement */
```

Dizilerin Fonksiyonlara Gönderilmesi

```
Effects of passing entire array by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8


Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[ 3 ] is 6
```

Çok Boyutlu Dizileri Fonksiyonlara Gönderme

- Tek boyutlu dizileri fonksiyona göndermekten farklı değildir.
- Her bir boyut için köşeli parantez kullanın, ilk boyut hariç diğerleri için büyüklük belirtin.
  - void writeMatrice (int [ ] [4], int rowNumber);
  - Bu tanımlama 4 sütuna sahip her matris için farklı satır numaralarına sahip olsalar da geçerli olur.
  - void writeMatrice (int [ ] [3] [4], int rowNumber);

Çok Boyutlu Dizileri Fonksiyonlara Gönderme

```
1  /* Fig. 6.21: fig06_21.c
2     Initializing multidimensional arrays */
3  #include <stdio.h>
4
5  void printArray( const int a[][ 3 ] ); /* function prototype */
6
7  /* function main begins program execution */
8  int main( void )
9  {
10    /* initialize array1, array2, array3 */
11    int array1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
12    int array2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5 };
13    int array3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };
14
15    printf( "Values in array1 by row are:\n" );
16    printArray( array1 );
17
18    printf( "Values in array2 by row are:\n" );
19    printArray( array2 );
20
21    printf( "Values in array3 by row are:\n" );
22    printArray( array3 );
23    return 0; /* indicates successful termination */
24 } /* end main */
```

Çok Boyutlu Dizileri Fonksiyonlara Gönderme

```
25
26  /* function to output array with two rows and three columns */
27  void printArray( const int a[][ 3 ] )
28  {
29     int i; /* row counter */
30     int j; /* column counter */
31
32     /* loop through rows */
33     for ( i = 0; i <= 1; i++ ) {
34
35        /* output column values */
36        for ( j = 0; j <= 2; j++ ) {
37           printf( "%d ", a[ i ][ j ] );
38        } /* end inner for */
39
40        printf( "\n" ); /* start new line of output */
41     } /* end outer for */
42  } /* end function printArray */
```

```
Values in array1 by row are:
1 2 3
4 5 6
Values in array2 by row are:
1 2 3
4 5 0
Values in array3 by row are:
1 2 0
4 0 0
```

Kaynaklar

- Doç. Dr. Fahri Vatansever, "Algoritma Geliştirme ve Programlamaya Giriş", Seçkin Yayıncılık, 12. Baskı, 2015.
- J. G. Brookshear, "Computer Science: An Overview 10th Ed.", Addison Wisley, 2009.
- Kaan Aslan, "A'dan Z'ye C Klavuzu 8. Basım", Pusula Yayıncılık, 2002.
- Paul J. Deitel, "C How to Program", Harvey Deitel.
- Bayram AKGÜL, C Programlama Ders notları