



BİLGİSAYAR PROGRAMLAMA 1

Ders Notu 9– Fonksiyonlar

Konya Teknik Üniversitesi
Elektrik – Elektronik Mühendisliği Bölümü

26.04.2024

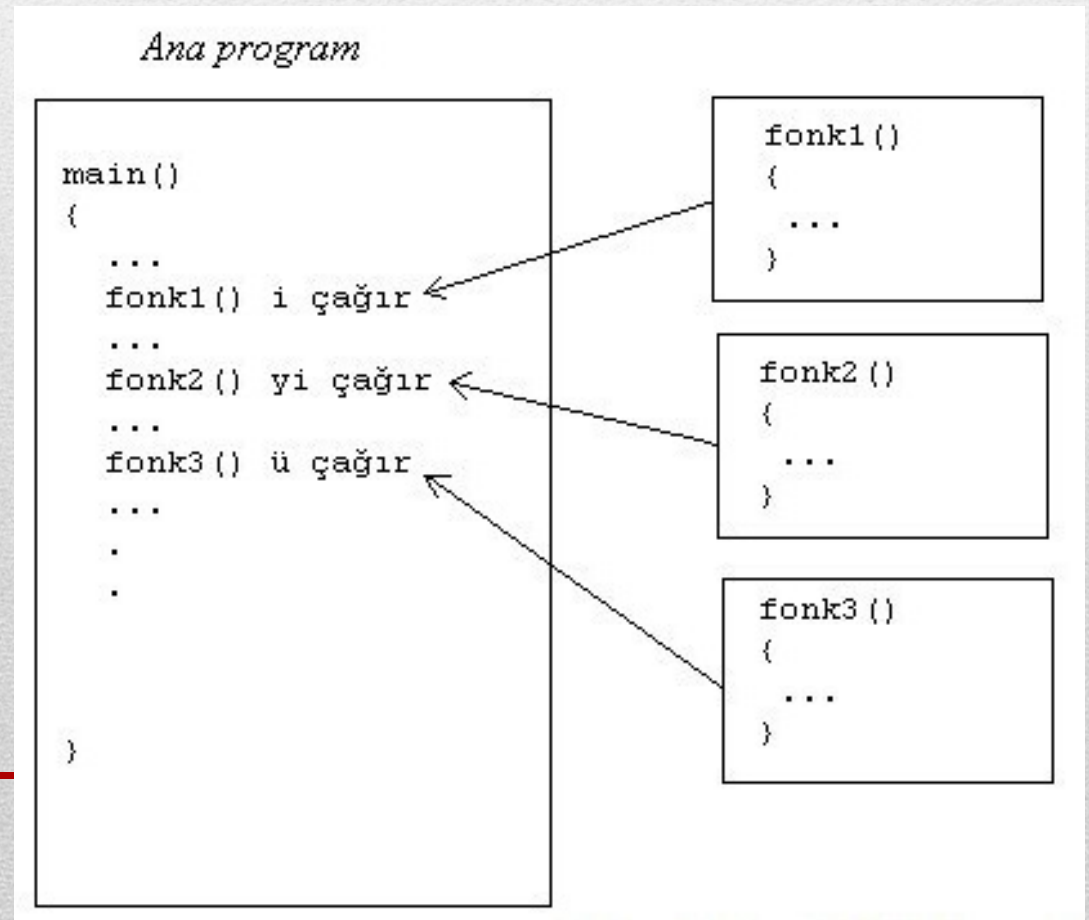
Konya

Fonksiyonlar

- C dilindeki fonksiyonları, girdisi ve çıktısı olan kapalı bir kutu gibi düşünebiliriz.
- Fonksiyonlar girdileri alır, bunlara göre işlemleri gerçekleştirirler ve isteğe bağlı olarak sonuç üretirler.
- Biz de fonksiyonların ürettiği bu çıktıları kullanırız.
- printf, scanf, sqrt, vb... birer fonksiyon örneğidir.
- Her bir fonksiyonu kendi işini yapan bir altprogram gibi düşünebiliriz.
- main() fonksiyonu da, program çalıştırıldığında otomatik olarak çağrılan bir fonksiyondur.

Neden Fonksiyonlara İhtiyaç Var?

- Geniş programları yazarken, küçük parçaları ya da her biri orijinal programdan daha kolay kullanılabilecek modülleri (daha önceden hazırlanmış program parçacıkları) birleştirmek kolaylık sağlar.
- Karmaşık yapıli programları sadeleştirir ve bu programları modüler bir hale getirir.
- Bazen standart kütüphanelerde tanımlı fonksiyonlar bütün ihtiyaçlarımızı karşılayamazlar.
- Fonksiyonlar, programcıların tekrarlanan kodlar yazmalarını önlerler.

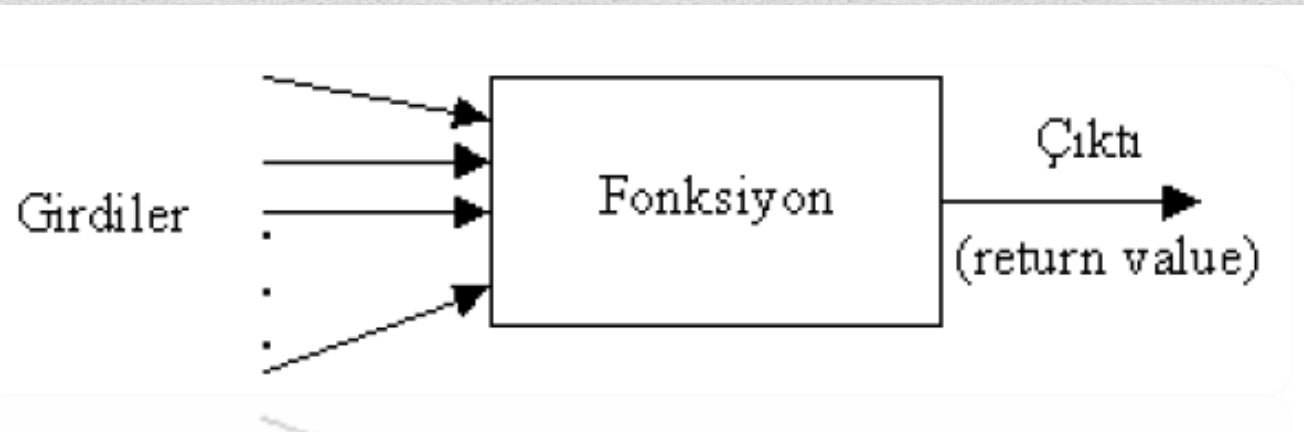


Fonksiyonlar

- Bir fonksiyon çağrıldığında;
 - Fonksiyon kendisine parametre olarak verilen veriler üzerinde çeşitli işlemler gerçekleştirir.
 - İstenildiği takdirde bu işlemlerin sonucunu çağrıldığı yere geri döndürür (geri yollar).

Örnek: toplamaYap fonksiyonu

- Girdiler: 2 ve 6 ise çıktı: 8
- Girdiler: 10 ve 15 ise çıktı: 25



Fonksiyonun Yapısı

- C'de fonksiyon bir veya daha fazla işlem satırından oluşan kodların bir kod bloğu şeklinde yapılandırılması ile oluşturulur. 3 Temel Parçadan oluşur:

dönüşVeriTipi fonksiyonAdı (argümanTürü);

//Fonksiyon bildirimi

main() {

//main Ana Program

fonksiyonAdı (argümanlar);

//Fonksiyonun çağırılması

}

dönüşVeriTipi fonksiyonAdı (argümanTürü ve adları)

//Fonksiyon gövdesi

{

yerel bildirimler;

işlem satırları

...

işlem satırları

return ifade;

}

Fonksiyonun Yapısı

```
dönüşVeriTipi fonksiyonAdı (argümanTürü ve adları)    //Fonksiyon gövdesi
{
    yerel bildirimler;                                // Yerel bildirimler/değişkenler sadece içinde
    işlem satırları                                    // bulundukları fonksiyonda kullanılabilir
    ...
    işlem satırları
    return ifade;
}
```

- **VeriTipi** : Fonksiyonun geri verdiği veri tipini gösterir.
- **fonksiyonAdı** : Fonksiyon adını gösterir.
- **argümanlar** : Fonksiyona geçirilen verileri gösterir. Argüman yerine Parametre veya Girdi ifadesi de kullanılmaktadır.
- **işlem satırı** : Fonksiyon içindeki işlem satırlarını gösterir.
- **return** : Verileri geri döndürmeye yarar. Son satırda kullanılması şart değildir.
- **ifade** : Değişken, sabit ve işlemciler kullanılabilir. Elde edilen veri türü, fonksiyonun geri döndürdüğü veri türü ile aynı olmalıdır.

Fonksiyonun Yapısı-Örnek

- Örnek 1 Toplama Yap Fonksiyonu:

```
ders9.1.c
1  #include <stdio.h>
2
3  int toplamaYap(int,int);          //Fonksiyon bildirimi
4
5
6  int main()                        // main Ana Program
7  {
8      int a,b,toplam;
9      printf ("Birinci sayiyi giriniz\n");
10     scanf ("%d",&a);
11     printf ("Ikinci sayiyi giriniz\n");
12     scanf ("%d",&b);
13
14     toplam=toplamaYap(a,b);        //Fonksiyonun çağırılması
15     printf("Iki sayinin toplami=%d",toplam);
16
17     return 0;
18 }
19
20 int toplamaYap(int sayi1,int sayi2) //Fonksiyon gövdesi
21 {
22     int sonuc;
23     sonuc=sayi1+sayi2;
24     return sonuc;
25 }
```

```
C:\Users\Kemal\Desktop\Google Drive\
Birinci sayiyi giriniz
2
Ikinci sayiyi giriniz
8
Iki sayinin toplami=10
-----
Process exited after 5.004 se
Press any key to continue . .
```


Return ve Veri Tipi İfadesi

- Bir fonksiyonun geri verdiği veri tipi **dizi** dışında herhangi bir tür olabilir.
 - Fonksiyon dizi tipinde bir veri geri döndüremez, ancak farklı yöntemlerle bu işlemi gerçekleştirebiliriz.
- veri-tipi **ifadesi**:
 - Eğer bir fonksiyonun adının başında veri türü tanımlanmazsa, fonksiyon int bir değer geri verir.
 - Eğer bir fonksiyonun int değer dışında bir veri türü geri vermesi isteniyorsa, fonksiyon adının başında mutlaka bir veri türü tanımlanması gerekir.

Return ve Veri Tipi İfadesi

- Örnekler:

return (a+b/c); /*parantez kullanmak zorunlu değil*/

return 10; /*değişken kullanmak mecbur değil*/

return topla(a,b)/2; /*önce topla fonksiyonu çalışır*/

- Programın çözüm mantığına göre bir fonksiyon içerisinde birden çok geri dönüş değeri kullanılabilir.
- Fakat, ilk karşılaşılan return deyiminden sonra fonksiyon sonlanır ve çağrılan yere bu değer gönderilir

Fonksiyon Bildirimi

- Bir veri türü geri veren bir fonksiyon kullanılacağı zaman, bu fonksiyonu kullanmadan önce, fonksiyonun geri vereceği veri türünün, fonksiyona geçirilecek argümanların sayısının ve türlerinin programın başında belirlenmesi gerekir. Bu işleme **Fonksiyon bildirimi** adı verilir.
- Fonksiyon bildiriminde geri döndürülecek veri türü belirtilmediği takdirde, bazı derleyiciler bu fonksiyonun geri vereceği değeri int olarak kabul eder.
- Eğer fonksiyonun gövdesi, main() fonksiyonundan önce yazılırsa, ayrıca prototip yöntemiyle bir bildirim yapılmasına ihtiyaç duyulmaz.

Fonksiyon Bildirimi- Örnek

- Örnek 2 Toplama Yap Fonksiyonu (float değişkeni kullanılarak):

```
1  #include <stdio.h>
2
3  //float toplamaYap(float,float);      //Fonksiyon bildirimi
4
5
6  int main()                          // main Ana Program
7  {
8      float a,b,toplam;
9      printf ("Birinci sayiyi giriniz\n");
10     scanf ("%f",&a);
11     printf ("Ikinci sayiyi giriniz\n");
12     scanf ("%f",&b);
13
14     toplam=toplamaYap(a,b);          //Fonksiyonun çağırılması
15     printf("İki sayinin toplami=%f",toplam);
16
17     return 0;
18 }
19
20 float toplamaYap(float sayi1,float sayi2)    //Fonksiyon gövdesi
21 {
22     float sonuc;
23     sonuc=sayi1+sayi2;
24     return sonuc;
25 }
```

Derleme Mesajları | Hata ayıkla | Arama sonuçları | Kapat

Mesaj
na\Deskto... [Error] conflicting types for 'toplamaYap'
na\Deskto... [Note] previous implicit declaration of 'toplamaYap' was here

Void (Boş) Fonksiyon

- Bir fonksiyon için geri dönen veri türü ve parametre tanımlanmayacaksa, bu ifadelerin yerine **void** ifadesini kullanarak bu durumu derleyiciye bildirebiliriz.
- `main()` fonksiyonundan önce de void ifadesini kullanılabilir.
- Parantez içine void yazılmaması, yani boş bırakılması da aynı anlama gelir.
- `void fonksiyonAdı (void)`
 {
 işlem satırları
 }

Fonksiyon Bildirimleri

Örnek	Açıklama
int islem();	Tam sayı değer dönen ve parametre girdisi içermeyen bir fonksiyon
int islem(void);	Tam sayı değer dönen ve parametre girdisi içermeyen bir fonksiyon
int islem(int x);	Tam sayı değer dönen ve tam sayı türünde parametre girdisi olan bir fonksiyon
void islem();	Değer dönmeyen ve parametre girdisi olmayan bir fonksiyon
void islem(int x);	Değer dönmeyen ve tam sayı türünde parametre girdisi olan bir fonksiyon

Void (Boş) Fonksiyon Örnek

- Örnek 3:

ders9.3.c

```
1  #include <stdio.h>
2
3  float toplamaYap(float,float);           //Fonksiyon bildirimleri
4  void herkesiSelamla();
5
6  int main()                             // main Ana Program
7  {
8      float a,b,toplam;
9      printf ("Birinci sayiyi giriniz\n");
10     scanf ("%f",&a);
11     printf ("Ikinci sayiyi giriniz\n");
12     scanf ("%f",&b);
13
14     toplam=toplamaYap(a,b);              //1. Fonksiyonun çağırılması
15     printf("Iki sayinin toplami=%f",toplam);
16
17     herkesiSelamla();                    //2. Fonksiyonun çağırılması
18
19     return 0;
20 }
21
22 float toplamaYap(float sayi1,float sayi2) //1. Fonksiyon gövdesi
23 {
24     float sonuc;
25     sonuc=sayi1+sayi2;
26     return sonuc;
27 }
28
29 void herkesiSelamla()                    //2. Fonksiyon gövdesi
30 {
31     printf("\n\n Merhaba Arkadaslar \n\n");
32 }
```

C:\Users\Kemal\Desktop\Google Drive\mas

```
Birinci sayiyi giriniz
2
Ikinci sayiyi giriniz
8
Iki sayinin toplami=10.000000

Merhaba Arkadaslar

-----
Process exited after 30.8 second
Press any key to continue . . .
```


İç içe Fonksiyonlar

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    mesaj();
    printf(" Hos geldiniz...\n");
    return 0;
}

void mesaj()
{
    printf("Algoritma");
    mesaj2();
}

void mesaj2()
{
    printf(" ve Programlama");
}
```

Algoritma ve Programlama Hos geldiniz...

Fonksiyonlara Parametre Aktarımı

- Fonksiyonlara parametre aktarımı iki şekilde yapılabilir:

1) Değer ile çağırma (Calling by Value)

Bu yöntemde değişkenlerin değerleri çağrılan fonksiyonun içine gönderilir. Bu değerler çağrılan fonksiyon içindeki değişkenlere atanarak işlemler yapılır. Fonksiyondaki işlemler tamamlandığında ana fonksiyondaki değişkenlerde herhangi bir değişiklik olmaz.

2) Adres ile çağırma (Calling by Reference)

Değişkenlerin hafızada tutulduğu adresler çağrılan fonksiyona gönderilir. Fonksiyondaki işlemler sonucunda ilgili adreslerdeki değerler güncelleneceğinden değişkenlerin değeri de değişmiş olur.

Fonksiyonlara Parametre Aktarımı

Değer ile çağırma (Calling by Value): Örnek 4

ders9.4.c

```
1  #include <stdio.h>;
2
3  float ortalama(int, int, int); //fonksiyon bildirimi
4
5  main()
6  {
7      int m,n,p;
8      float k;
9      printf("\nBirinci sayiyi giriniz:");
10     scanf("%d",&m);
11     printf("\nIkinci sayiyi giriniz:");
12     scanf("%d",&n);
13     printf("\nUcuncu sayiyi giriniz:");
14     scanf("%d",&p);
15     k=ortalama(m,n,p);           //fonksiyonun çağırılması ve fonksiyondan
16     //geri gelecek sonuç değerin k değişkenine atanması
17
18     printf("\nHesaplanan Ortalama:%f",k);
19 }
20 float ortalama(int a, int b,int c) //fonksiyonun tanımlanması
21 {
22     float g;
23     g=(a+b+c)/3.0;
24     return g;
25 }
```

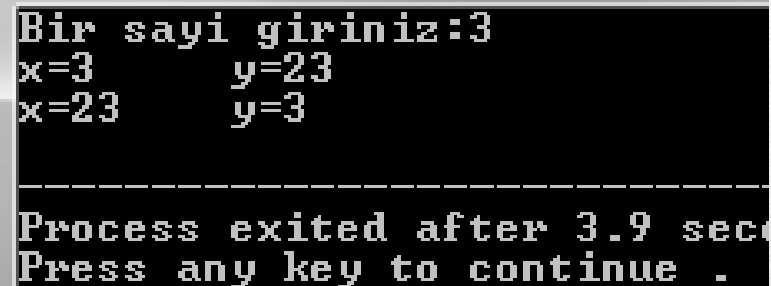
C:\Users\Kemal\Desktop\Google Drive\masaustu\

```
Birinci sayiyi giriniz:3
Ikinci sayiyi giriniz:5
Ucuncu sayiyi giriniz:6
Hesaplanan Ortalama:4.666667
-----
Process exited after 4.917 seconds w
Press any key to continue . . .
```

Fonksiyonlara Parametre Aktarımı

Adres ile çağırma (Calling by Reference): Örnek 5

```
1  #include <stdio.h>;
2
3  int yerdegistir(int*, int*);
4
5  main()
6  {
7      int x,y;
8      printf("Bir sayi giriniz:");
9      scanf("%d",&x);
10     y=23;
11     printf("x=%d    y=%d \n",x,y); //x ve y'nin ilk hali
12     yerdegistir(&x,&y);
13     printf("x=%d    y=%d \n",x,y); //x ve y'nin son hali
14 }
15 int yerdegistir(int *a, int *b)
16 {
17     int g;
18     g=*a; //a'nın işaret ettiği yerdeki değer g'ye atanıyor
19     *a=*b; //b'nin işaret ettiği yerdeki değer a'ya atanıyor
20     *b=g; //g'de tutulan değer(a'nın ilk değeri) b'ye atanıyor
21
22 }
```



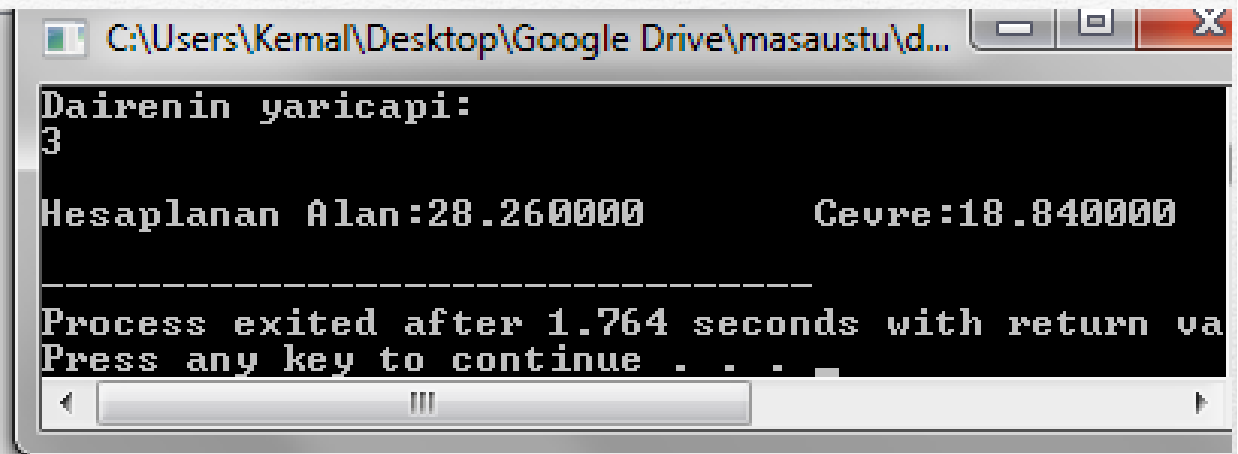
```
Bir sayi giriniz:3
x=3      y=23
x=23     y=3

-----
Process exited after 3.9 seconds
Press any key to continue .
```


Fonksiyonlara Parametre Aktarımı

Karma Kullanım: Örnek 6

```
1  #include <stdio.h>;
2
3  void daire(float, float*, float*);
4
5  main()
6  {
7      float yaricap, alan, cevre;
8      puts("Dairenin yaricapi:");
9      scanf("%f",&yaricap);
10
11     daire(yaricap, &alan, &cevre);
12     printf("\nHesaplanan Alan:%f   Cevre:%f\n",alan,cevre);
13 }
14 void daire(float r, float *al, float*cev)
15 {
16     *al=3.14*r*r;
17     *cev=2*3.14*r;
18     r=r+1;
19
20 }
```



```
C:\Users\Kemal\Desktop\Google Drive\masaustu\d...
Dairenin yaricapi:
3
Hesaplanan Alan:28.260000      Cevre:18.840000
-----
Process exited after 1.764 seconds with return va
Press any key to continue . . .
```

Fonksiyonlara Parametre Aktarımı

Karma Kullanım (yarı çap değerinin değişmemesi): Örnek 7

```
1 #include <stdio.h>;
2
3 void daire(float, float*, float*);
4
5 main()
6 {
7     float yaricap, alan, cevre;
8     puts("Dairenin yaricapi:");
9     scanf("%f",&yaricap);
10
11     daire(yaricap, &alan, &cevre);
12     printf("\nHesaplanan Alan:%f      Cevre:%f\n",alan,cevre);
13
14     printf("\nYaricapin son degeri:%f \n",yaricap);
15 }
16 void daire(float r, float *al, float*cev)
17 {
18     *al=3.14*r*r;
19     *cev=2*3.14*r;
20     r=r+1;
21
22 }
```

C:\Users\Kemal\Desktop\Google Drive\masaustu\ders için örnek dol

Dairenin yaricapi:
3

Hesaplanan Alan:28.260000 Cevre:18.840000

Yaricapin son degeri:3.000000

Process exited after 8.772 seconds with return v
Press any key to continue . . .

Fonksiyonlara Parametre Aktarımı

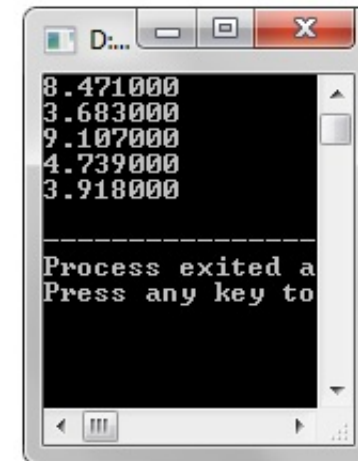
- Fonksiyon kullanımı planlanırken değer ile veya adresi ile çağırmanın hangisinin kullanılacağı fonksiyonun yapacağı işleme ve çağırana döndüreceği parametrelerin sayısına göre seçilmelidir.
- Çağırana tek bir sonuç döndürülecekse değer ile çağırma kullanılarak return ile değer döndürülmesi uygun olabilir.
- Ancak birden fazla değer döndürmek gerekiyorsa return kullanımı yetersiz kalabileceğinden dolayı adres ile çağırma yöntemi kullanılmalı ve değişkenlerin değerleri güncellenerek fonksiyondaki işlemlerin sonuçları çağıran programa döndürülmelidir.

Dizilerin Fonksiyonlarda Kullanımı

- Diziler de diğer değişkenler gibi fonksiyonlara gönderilip işleme tabi tutulabilir. Örnek 8:

ders9.8.c

```
1  #include <stdio.h>
2
3  void dizi_yazdir(float x[], int  );
4
5  int main()
6  {
7      float dizi[5]={8.471, 3.683, 9.107, 4.739, 3.918};
8      dizi_yazdir(dizi, 5);
9      return 0;
10 }
11
12 void dizi_yazdir(float x[], int n)
13 {
14     int i;
15     for(i=0;i<n;i++)
16     {
17         printf("%f \n",x[i]);
18     }
19
20 }
21
```



Özyinelemeli (Recursive) Fonksiyonlar

- Belirli bir dönüş kriteri gelene kadar kendi kendini çağıran fonksiyonlara özyinelemeli (recursive) fonksiyon denir.

```
#include <stdlib.h>

int faktoryel_hesapla(int sayi);

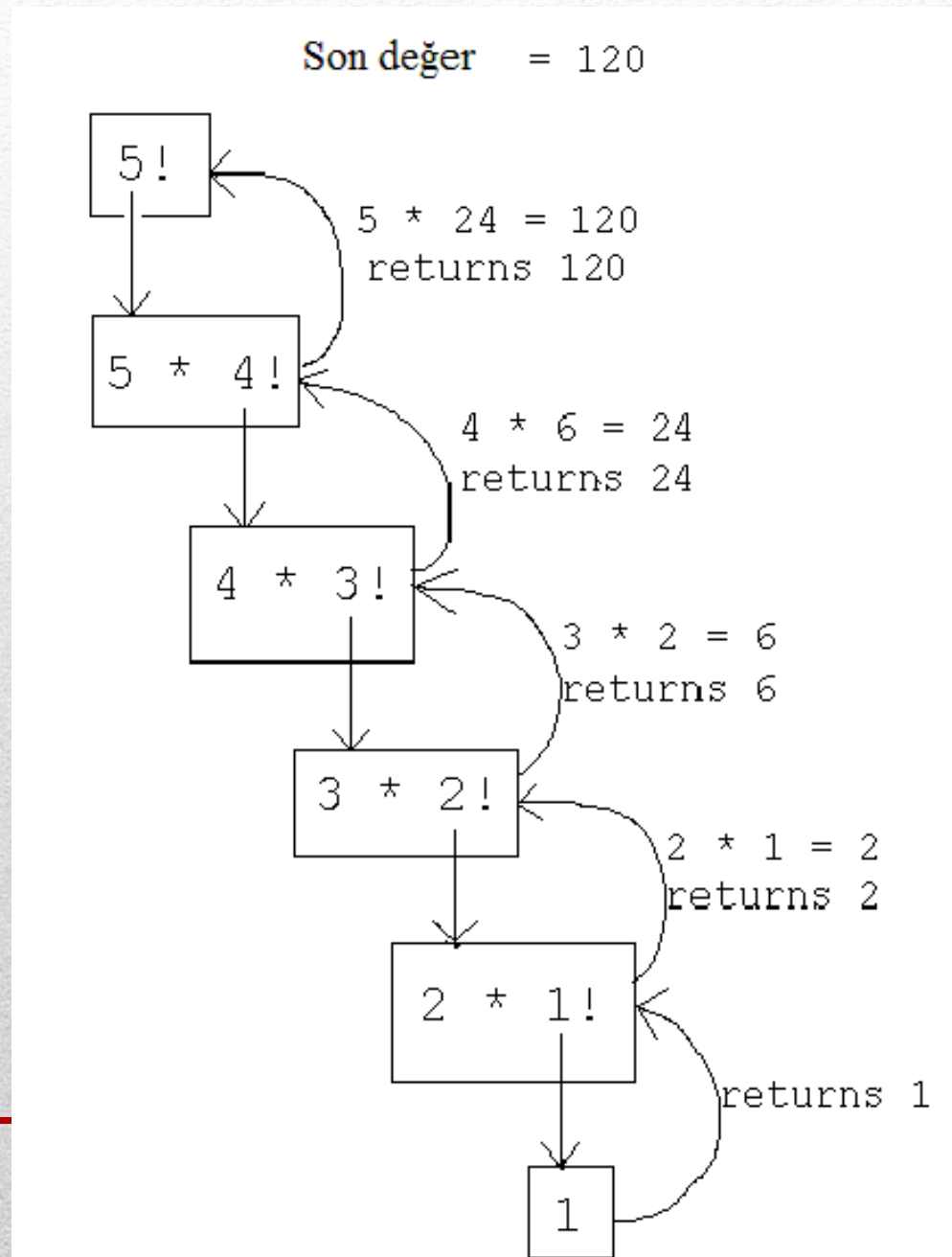
int main()
{
    int sayi;
    printf("Faktoryel hesabi icin sayi giriniz:");
    scanf("%d", &sayi);

    printf("Faktoryel sonucu: %d \n", faktoryel_hesapla(sayi));
    return 0;
}

int faktoryel_hesapla(int sayi)
{
    if (sayi <= 1)
        return 1;
    else
        return (sayi * faktoryel_hesapla(sayi-1)); //yineleme kismi
}
```

```
Faktoryel hesabi icin sayi giriniz:5
Faktoryel sonucu: 120
```

Özyinelemeli (Recursive) Fonksiyonlar



Özyineleme vs Tekrar

- Tekrar ve yinelemenin ikisi de döngü içerir.
- Tekrar özellikle döngü yapısını kullanırken, yineleme döngüyü fonksiyon çağrılarının tekrarında kullanır.
- Tekrar ve yinelemenin ikisi de bir sonlandırma testi içerirler.
- Yineleme temel bir durumla karşılaşıldığında, tekrar ise döngü devam koşulu yanlış hale geldiğinde sona erer.
- Yineleme bir çok negatif özelliğe sahiptir.
- Yineleme, mekanizmayı sürekli çağırarak fonksiyon çağrılarının artmasına sebep olur. Bu, işlemci için fazladan yük demektir.

Özyineleme vs Tekrar

- Ayrıca her yineleme çağrısı, fonksiyonun başka bir kopyasının oluşmasına sebep olur, bu da hafızayı fazladan işgal etmek demektir.
- Tekrarlı döngüler sayesinde, fonksiyonların sürekli olarak çağrılması ve fazladan hafıza kullanılması engellenir.
- Yinelemeli olarak çözülen her problem, tekrarlı bir biçimde çözülebilir. Yineleme yaklaşımı genelde problemi daha iyi yansıttığı ve daha kolay anlaşılan ve hataları kolay ayıklanan programlar yazılmasını sağlattığı için tekrar yaklaşımına göre bazen tercih edilebilir.
- Yinelemeli çözümleri seçmenin başka bir sebebi de tekrarlı çözümün kolaylıkla bulunamayışıdır.

Fonksiyon Kullanım Hataları

- Fonksiyon tanımlamalarında geri dönüş değerini unutmak.
- Geri dönüş tipi void olarak bildirilmiş bir fonksiyonun bir değer geri döndürmesi bir yazım hatasıdır.
- Aynı tipte fonksiyon parametrelerini *double x*, *double y* yerine *double x, y* olarak bildirmek. *double x, y* biçiminde parametre bildirmek, *y parametresinin tipinin int olmasına* sebep olur. Çünkü belirtilmeyen parametre tipi bazı derleyicilerde otomatik olarak int tipinde varsayılır.

Fonksiyon Kullanım Hataları

- Parametre (Argüman) listesini yazdığımız parantezlerin dışına noktalı virgül koymak yazım hatasıdır.
- Bir fonksiyon parametresini daha sonradan fonksiyon içinde yerel bir değişken olarak kullanmak bir yazım hatasıdır.
- Bir fonksiyon içinde başka bir fonksiyon tanımlamak yazım hatasıdır.
- Fonksiyon bildiriminin (prototipinin) sonuna noktalı virgül koymamak bir yazım hatasıdır.

Tavsiyeler

- Birden fazla fonksiyon kullanılan programlarda, **main** fonksiyonu programın esas görevini yerine getiren fonksiyonların çağırıcısı olarak kullanılmalıdır.
- Her fonksiyon, iyi olarak tanımlanmış tek bir işi yapacak şekilde sınırlandırılmalıdır ve fonksiyon ismi, fonksiyonun görevini etkili bir biçimde açıklamalıdır. Bu, özetlemeyi ve yazılımın yeniden kullanılabilirliğini sağlar.
- Eğer fonksiyonun görevini açıklayacak etkili bir isim bulamıyorsanız muhtemelen yazdığınız fonksiyon birden fazla görevi yerine getirmeye çalışmaktadır. Bu tarzda fonksiyonları daha küçük fonksiyonlara bölmek en iyi yoldur.

Tavsiyeler

- Bir fonksiyon genellikle bir sayfadan daha uzun olmamalıdır. Küçük fonksiyonlar yazılımın yeniden kullanılabilmesini sağlar.
- Programlar, küçük fonksiyonların bir araya getirilmesiyle yazılmalıdır. Bu, programların daha kolay yazılması, değiştirilmesi ve hatalarının giderilmesini sağlar.
- Çok fazla sayıda parametreye ihtiyaç duyan fonksiyonlar birden fazla görevi yerine getiriyor olabilir. Böyle fonksiyonları ayrı görevleri gerçekleştiren daha küçük fonksiyonlara bölmek gerekir.
- Fonksiyonun başlığı mümkünse bir satıra sığmalıdır.

Ödev

- Hazırlayacağınız program kullanıcıdan 5 öğrencinin bir dersten aldıkları vize ve final notlarını isteyecektir. Bu notlara göre yıl sonu notunu hesaplayıp harf karşılıklarını ekrana bastıracaksınız.
- 2 farklı yöntem kullanacaksınız:
 - 1. Yöntem: Kullanıcıdan istediğiniz vize ve final notları girildiğinde bunları harfe dönüştüren bir fonksiyona gönderip ekrana harf karşılıklarını basacaksınız daha sonra kullanıcıdan diğer öğrencinin notunu girmesini isteyeceksiniz.
 - 2. yöntem: Kullanıcıdan tüm öğrencilerin vize notlarını peş peşe girmesini isteyecek ve bunları bir dizide saklayacaksınız. Final için de aynı işlemi uygulayacaksınız. Sonra bu iki dizideki değerleri harfe dönüştürme fonksiyonuna gönderip her öğrencinin harf notunu peş peşe ekrana bastıracaksınız

Kaynaklar

- Programlama Sanatı, Algoritmalar, C Dili Uyarlaması, Dr. Rifat ÇÖLKESEN, Papatya Yayıncılık
- Her Yönüyle C, Tefvik KIZILÖREN, Kodlab
- C Programlama Dili, Dr. Rifat ÇÖLKESEN, Papatya Yayıncılık
- Celal Bayar Üniversitesi, Hasan Ferdi Turgutlu Teknoloji Fakültesi, YZM1105 Ders Notu
- https://www.bilgigunlugum.net/prog/cprog/c_fonksiyon (Erişim Tarihi: 13.03.2020)