CENG 111 ALGORİTMALAR VE PROGRAMLAMA

Doç. Dr. Tufan TURACI
tturaci@pau.edu.tr

- Pamukkale Üniversitesi
- Mühendislik Fakültesi
- Bilgisayar Mühendisliği Bölümü

- Hafta 11
- 29 Kasım 2022

FONKSİYONLAR

- Fonksiyon tanımı
- C dilinde fonksiyonlar
- Fonksiyonları çağırmak
  - Fonksiyonları değer ile çağırmak
  - Fonksiyonları adres ile çağırmak(İşaretçiler konusunda anlatılacaktır…)
- Yinelemeli(Rekürsif) Fonksiyonlar

Fonksiyon tanımı

• Gerçek hayatta gördüğümüz yazılım problemleri uzun ve karmaşık yapıdadırlar.

• Şu ana kadar kısa programlama örnekleri yaptık.

• Uzun ve karmaşık yapıdaki programları yazmanın kolay bir yolu problemleri küçük parçalara ayırıp daha sonra bu parçaları birleştirerek problem çözmeye dayalıdır.

• Fonksiyonlar, karmaşık yapılı programların anlaşılır bir şekilde yazılması ve bu programları daha efektif bir şekilde yazmak için kullanılırlar.

•Fonksiyonlar, tekrar eden kod parçalarının yazılmasını engeller.

•Fonksiyonlar, başka bir fonksiyondan isimleri ile çağırılarak çalışırlar.

C dilinde fonksiyonlar

- C dili fonksiyonlar üzerine kuruludur.
- main() fonksiyonu C programının ana fonksiyonudur.
- main() fonksiyonun çalışması için başka bir fonksiyon tarafından çağırılması gerekmez.
- Şu ana kadar yazdığımız programlarda bir tek main() fonksiyonu vardı, buradan anlaşıldığı gibi main() fonksiyonu her C programında bulunmalıdır.
- main() fonksiyonundan bir çok farklı fonksiyon çağırılabilir.

- Bir fonksiyondan farklı bir fonksiyon çağırılabilir.

```
main ( )
{…
fonksiyon_A ( ); // main fonksiyonun içinde A fonksiyonu çağırıldı.
}

fonksiyon_A ( )
{

fonksiyon_B( ) ; // A fonksiyonun içinde B fonksiyonu çağırıldı.

}

fonksiyon_B ( )
{
….
}
```

-- Fonksiyonlar, belirli sayıda verileri kullanır, bu verileri işler ve bunun sonucunda bir sonuç üreten programlardır (komut grubudur).

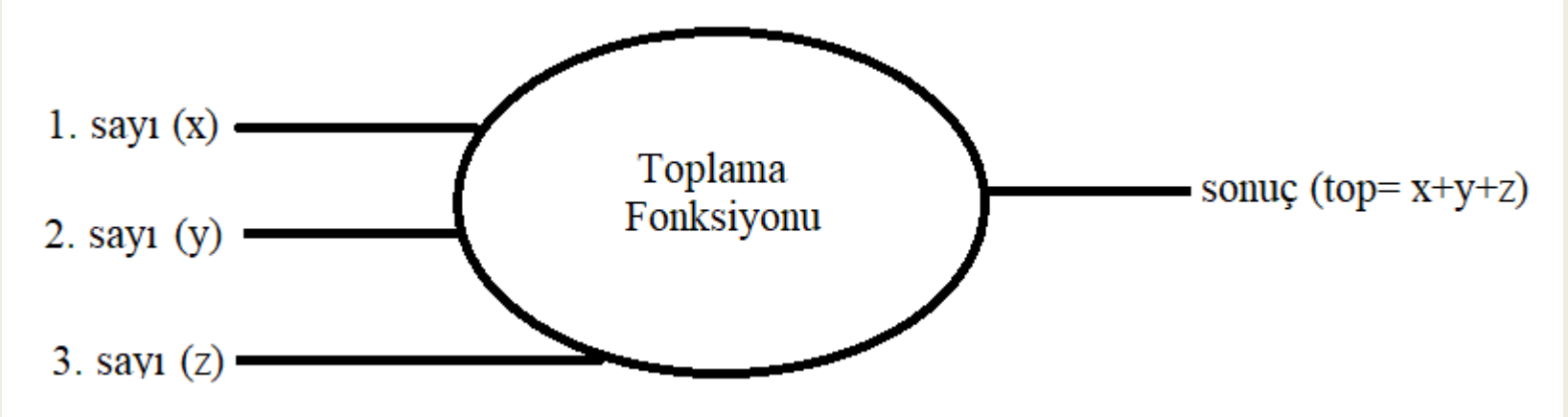
-- Her fonksiyonun bir adı, ve fonksiyona gelen değerleri gösteren parametre değerleri vardır.

-- Bir fonksiyon bu parametreleri ele alır ve sonuç olarak bir değer bulur.

-- Bu sonuç ya fonksiyonda yazdırılır ya da fonksiyonun çağırıldığı yere geri döndürülür (return sonuç).

-- Bir fonksiyonun kaç parametre girişi olursa olsun, sadece bir çıkışı vardır.

Örnek: Klavyeden girilen 3 sayının toplamını bir fonksiyonda bulan ve sonucu Çağırıldığı fonksiyonda yazdıran bir C programı yazınız.



Fonksiyon tipi: int

Fonksiyon adı: toplama

Parametreler: x, y ve z

Geri dönüş değeri: top=x+y+z

1. yol:

```
#include<stdio.h>
#include<conio.h>
int toplama(int a,int b,int c); // fonksiyon main() den sonra kullanılacaksa, main() önce tanımlanmalıdır.
```

Fonksiyon protitipi olarak adlandırılır.

```
int main()  // ana fonksiyon
{ int x,y,z,sonuc;
printf("x degerini giriniz: ");
scanf("%d",&x);
printf("y degerini giriniz: ");
scanf("%d",&y);
printf("z degerini giriniz: ");
scanf("%d",&z);
sonuc=toplama(x,y,z);
printf("uc sayinin toplami= %d",sonuc);
getch ();
return 0;
}
int toplama(int a,int b,int c) // fonksiyon üç sayının toplamını bulur.
{ int top;
top=a+b+c;
return top;
}
```

```
x degerini giriniz: 3
y degerini giriniz: 5
z degerini giriniz: 7
uc sayinin toplami= 15
--------------------------------
```

2. yol:

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

int toplama(int a,int b,int c)  // fonksiyon, main() den önce yazıldı…
{ int top;
top=a+b+c;
return top;
}

int main()
{ int x,y,z,sonuc;
printf("x degerini giriniz: ");
scanf("%d",&x);
printf("y degerini giriniz: ");
scanf("%d",&y);
printf("z degerini giriniz: ");
scanf("%d",&z);
sonuc=toplama(x,y,z);
printf("uc sayinin toplami= %d",sonuc);
getch ();
return 0;
}
```

```
x degerini giriniz: 4
y degerini giriniz: 6
z degerini giriniz: 8
uc sayinin toplami= 18
--------------------------------
```

1. ve 2. yolda sonuç ana programda yazıldı.

Sonuç toplama() fonksiyonunda yazdırıldı…

```
#include<stdio.h>
#include<conio.h>
void toplama(int a,int b,int c);

int main()
{ int x,y,z;
printf("x degerini giriniz: ");
scanf("%d",&x);
printf("y degerini giriniz: ");
scanf("%d",&y);
printf("z degerini giriniz: ");
scanf("%d",&z);
toplama(x,y,z); // toplama fonksiyonu çağırıldı, sonuç toplama fonksiyonunda yazdırıldı ve program sonlandırıldı.
getch ();
return 0;
}

void toplama(int a,int b,int c)
{ int top;
top=a+b+c;
printf("uc sayinin toplami= %d",top); // Sonuç burada yazdırıldı…
}
```

```
x degerini giriniz: 5
y degerini giriniz: 10
z degerini giriniz: 15
uc sayinin toplami= 30
--------------------------------
```

Parametre ve Argüman

---Fonksiyon çağrılırken gönderilen değerlere **Argüman** denir.

---Fonksiyon bildiriminde, fonksiyona girdi olarak, kullanılan değişkenlere **Parametre** denir.

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

int toplama(int a,int b,int c);
int main()  // ana fonksiyon
{ int x,y,z,sonuc;
printf("x degerini giriniz: ");
scanf("%d",&x);
printf("y degerini giriniz: ");
scanf("%d",&y);
printf("z degerini giriniz: ");
scanf("%d",&z);
sonuc=toplama(x,y,z);
printf("uc sayinin toplami= %d",sonuc);
getch ();
return 0;
}
```

```
int toplama(int a,int b,int c)
{ int top;
top=a+b+c;
return top;
}
```

Argüman: x,y,z

Parametre: a,b,c

Fonksiyon Bildirim Tipleri

int fonk_A();	Parametre içermeyen tam sayı değer döndüren bir fonksiyon
int fonk_A(void);	Parametre içermeyen tam sayı değer döndüren bir fonksiyon
int fonk_A(**int** x);	Tam sayı değer döndüren ve bir tam sayı parametresi girdisi olan bir fonksiyon
void fonk_A();	Değer dönmeyen ve parametre girdisi olmayan bir fonksiyon
void fonk_A(**int** x);	Tam sayı türünde parametre girdisi olan ve değer döndürmeyen bir fonksiyon

Fonksiyon Geri Dönüş Değerleri

--- Çağırılan bir fonksiyondan geri dönüş değeri **return** kullanılarak yapılır.

--- **return** deyimi iki önemli işlevi vardır:

*** fonksiyonun geri dönüş değerini oluşturur.

*** fonksiyonu sonlandırır.

Örnek:

```
return top;
return top/2;
return 10;
return topla(a,b,c)/3;
```

--- Programın çözüm mantığına göre bir fonksiyon içerisinde birden çok **return** kullanılabilir.

--- Fakat, ilk karşılaşılan **return** deyiminden sonra fonksiyon sonlanır ve çağrılan yere bu değer gönderilir.

void Fonksiyonlar

--- Bir fonksiyonun her zaman geri dönüş değeri olmayabilir.

--- Bu durumda **return** deyimi kullanılmayabilir. Eğer **return** deyimi yoksa, fonksiyon ana bloğu bitince kendiliğinden sonlanır.

--- Bu tipte fonksiyonların tipi **void** (boş, hükümsüz) olarak belirtilmelidir.

--- Geri dönüş değeri olmadığından çağırıldığı program bloğu içinde bir değişkene void fonksiyon atanamaz.

--- void fonksiyonlara parametre aktarımı yapılabilir.

Örnek 1: Değer dönmeyen ve parametre girdisi olmayan bir fonksiyona örnek.

```
#include<stdio.h>
#include<conio.h>
void fonk_A();  //  fonksiyon prototipi…

int main()
{
fonk_A();
printf(" Universitesi");
getch ();
return 0;
}

void fonk_A()
{ printf(" Pamukkale");
}
```

```
 Pamukkale Universitesi
--------------------------------
Process exited after 1.268 seconds with return value 0
Press any key to continue . . .
```

Örnek 2: Değer dönmeyen ve parametre girdisi olmayan iç içe çağırılan fonksiyonlara örnek.

```
#include<stdio.h>
#include<conio.h>
void fonk_A();
void fonk_B();

int main()
{
fonk_A();
printf(" Denizli");
getch ();
return 0;
}
void fonk_A()
{ fonk_B();
printf(" Universitesi");
}
void fonk_B()
{ printf(" Pamukkale");
}
```

```
 Pamukkale Universitesi Denizli
--------------------------------
Process exited after 11.58 seconds with return value 0
Press any key to continue . . .
```

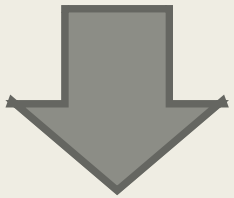
Örnek 3: Tam sayı türünde parametre girdisi olan ve değer döndürmeyen bir fonksiyon örneği. Klavyeden girilen bir n tamsayısına göre bir fonksiyonda n*n tipinde rastgele sayılardan oluşmuş bir matris üreten C programı yazınız.

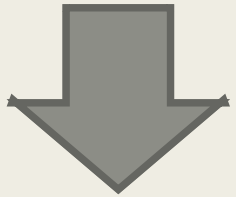
```
Bir pozitif tamsayi giriniz: 3
3 * 3 tipindeki matris, matris isimli fonksiyonda uretilmistir...
   3   9  14
   6   7  12
   4   4  20
```

```
Bir pozitif tamsayi giriniz: 8
8 * 8 tipindeki matris, matris isimli fonksiyonda uretilmistir...
  14  12  11  19   3  16   9   5
  13   8  16  17   7  15  10   9
   1   6  14   3  17  18   8  17
   9  17  15  10  20  13  11   5
  14   3  19  20  16  15  12  13
   7   9   2  16  19   4  18  18
  15  17   2  16  12  18   5  18
  19   7  14   1   6  20  18   7
```

C kodu:

```
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>
#include <time.h>
void matris(int a);

int main()
{ int n;
printf("Bir pozitif tamsayi giriniz: ");
scanf("%d",&n);
printf ("%d * %d tipindeki matris, matris isimli
fonksiyonda uretilmistir...\n",n,n);
matris(n);
getch ();
return 0;
}
```





```
void matris(int a)
{ int i,j,A[a][a],x;
  srand(time(NULL));
for(i=0;i<a;i++)
{
for (j=0;j<a;j++)
{x=1+(rand()%20);
  A[i][j]=x;}
}

for(i=0;i<a;i++)
{
for (j=0;j<a;j++)
{printf("%4d",A[i][j]);}
printf ("\n");
}

}
```

Örnek 4: Tam sayı türünde parametre girdisi olan ve değer döndürmeyen bir fonksiyon örneği. Klavyeden üç öğrencinin notları giriliyor. Notları harf_notu fonksiyonuna gönderip notun harf notu karşılığını bulan bir C programı yazınız.

```
1. ogrencinin notunu giriniz: 97
1. ogrencinin notu= 97 ve harf notu= A1
2. ogrencinin notunu giriniz: 58
2. ogrencinin notu= 58 ve harf notu= D1
3. ogrencinin notunu giriniz: 86
3. ogrencinin notu= 86 ve harf notu= A3


--------------------------------
```

C kodu:

```
#include<stdio.h>
#include<conio.h>
void harf_notu(int a);

int main()
{ int n,i,x;
for (i=0;i<3;i++)
{
printf ("%d. ogrencinin notunu giriniz: ",i+1);
scanf("%d",&x);
printf ("%d. ogrencinin notu= %d ve harf notu= ",i+1,x);
harf_notu(x);
printf("\n");
}

getch ();
return 0;
}
```

```
void harf_notu(int a)
{
if ((a>=95) && (a<=100)) printf("A1");
if ((a>=90) && (a<95)) printf("A2");
if ((a>=85) && (a<90)) printf("A3");
if ((a>=80) && (a<85)) printf("B1");
if ((a>=75) && (a<80)) printf("B2");
if ((a>=70) && (a<75)) printf("B3");
if ((a>=65) && (a<70)) printf("C1");
if ((a>=60) && (a<65)) printf("C2");
if ((a>=55) && (a<60)) printf("D1");
if ((a>=50) && (a<55)) printf("D2");
if ((a>=0) && (a<50)) printf("F1");
}
```

Örnek 5: Klavyeden girilen n ve r değerleri için kombinasyon C(n,r) ve Permütasyon P(n,r) degerlerini hesaplayan bir C programı yazınız.

C(n,r)=n!/((n-r)!*r!) ve P(n,r)=n!/(n-r)!
Faktoriyel bir fonksiyonsa hesaplanacaktır.

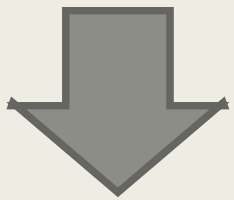
```
n degerini giriniz:6
r degerini giriniz:2
Kombinasyon degeri=15
Permutasyon degeri=30
--------------------------------
```

```
n degerini giriniz:5
r degerini giriniz:1
Kombinasyon degeri=5
Permutasyon degeri=5
--------------------------------
```

```
n degerini giriniz:10
r degerini giriniz:3
Kombinasyon degeri=120
Permutasyon degeri=720
--------------------------------
```

C kodu:

```
#include<stdio.h>
#include <conio.h>
int fakt(int x);

int main ()
{int n,r,comb,perm;
printf ("n degerini giriniz:");
scanf ("%d",&n);
printf ("r degerini giriniz:");
scanf ("%d",&r);

comb=fakt(n)/(fakt(n-r)*fakt(r));
printf ("Kombinasyon degeri=%d\n",comb);

perm=fakt(n)/fakt(n-r);
printf ("Permutasyon degeri=%d",perm);

getch();
return 0;
}
```



```
// fonksiyon yazımı...
// iki tane return kullandık. return kullanımına örnek
int fakt(int x)
{int i,faktoriyel,carp=1;
if ((x==0) || (x==1)) {return 1;}
    else
        {for (i=1;i<=x;i++)
          {carp=carp*i;}
         return carp;
        }
}
```

Fonksiyon Kullanım Hataları

---Fonksiyon tanımlamalarında geri dönüş değerini unutmak programın çalışmasında bir hataya sebep verebilir.

---Geri dönüş tipi **void** olarak bildirilmiş bir fonksiyonun bir değer geri döndürmesi bir yazım hatası olarak bilinmektedir.

---Aynı tipte fonksiyon parametrelerini ***float*** *a*, ***float*** *b* yerine ***float*** *a*, *b* olarak bildirirsek **b** parametresinin **int** tipinde olmasına sebep olur. Çünkü belirtilmeyen parametre tipi otomatik olarak **int** tipinde varsayılmaktadır.

--- Her bir parametrenin tipini ayrı ayrı belirtmek daha uygundur.

Fonksiyon Kullanım Hataları

--- Parametre 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 prototipinin sonuna noktalı virgül koymamak bir yazım hatasıdır.

Dizileri bir Fonksiyona Göndermek

---Diziler de değişkenler gibi bir fonksiyona parametre olarak aktarılabilirler.

--- Dizileri aktarma kuralı biraz farklıdır.

--- Her zaman dizinin yanında dizinin boyutu da aktarılmalıdır.

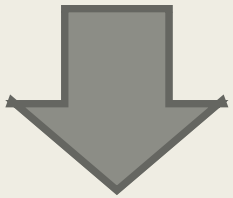
Örnek 1: Ana programdan 5 adet sayı giriliyor ve bu sayılar A isimli bir diziye aktarılıyor. A isimli dizideki elemanlar ortalama isimli bir fonksiyona gönderiliyor ve bu fonksiyon da dizinin elemanlarının ortalamasını bulunuyor. Bulunan ortalama değerini ana programda yazdıran bir C programı yazınız.

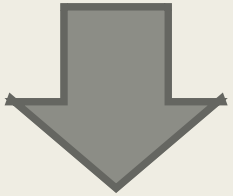
```
1. elemani giriniz:  45
2. elemani giriniz:  20
3. elemani giriniz:  35
4. elemani giriniz:  78
5. elemani giriniz:  90
Sonuc ana programda yazdiriliyor...
A nin degerlerinin ortalamasi= 53.60
--------------------------------
```

C kodu:

1. yol: Dizi ve boyutu ayrı ayrı gönderiliyor…

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

float ortalama(float B[ ], int n);
int main()
{ int i; float sonuc,A[5];
  for (i=0;i<5;i++)
  {printf ("%d. elemani giriniz:  ",i+1);
   scanf ("%f", &A[i]);
  }
  sonuc=ortalama(A,5);
  printf ("Sonuc ana programda yazdiriliyor...\n");
  printf ("A nin degerlerinin ortalamasi= %.2f",sonuc);
  getch();
  return 0;
}
```

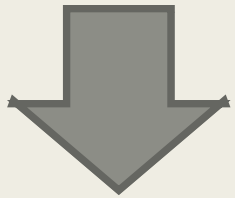


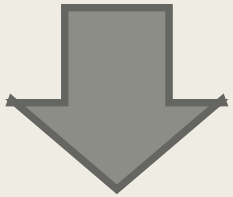


```
float ortalama(float B[ ], int n)
{int i; float ort, top=0.0;

 for (i=0;i<n;i++)
 {top=top+B[i];
 }
  ort=top/n;
  return ort;
}
```

2. yol: Dizi ve boyutu aynı anda gönderiliyor…

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

float ortalama(float B[5]);
int main()
{ int i; float sonuc,A[5];
  for (i=0;i<5;i++)
  {printf ("%d. elemani giriniz:  ",i+1);
   scanf ("%f", &A[i]);
  }
  sonuc=ortalama(A);
  printf ("Sonuc ana programda yazdiriliyor...\n");
  printf ("A nin degerlerinin ortalamasi= %.2f",sonuc);
  getch();
  return 0;
}
```





```
float ortalama(float B[5])
{int i; float ort, top=0.0;

 for (i=0;i<5;i++)
 {top=top+B[i];
 }
  ort=top/5.0;
  return ort;
}
```

3. yol: Ortalama değeri fonksiyonda yazdırılıyor.

```
#include <stdio.h>
#include <conio.h>
void  ortalama(float B[ ], int n);

int main()
{ int i; float A[5];
  for (i=0;i<5;i++)
  {printf ("%d. elemani giriniz:  ",i+1);
   scanf ("%f", &A[i]);
  }
  ortalama(A,5);
  getch();
  return 0;
}
```

```
void ortalama(float B[ ], int n)
{int i; float ort, top=0.0;

 for (i=0;i<n;i++)
 {top=top+B[i];
 }
  ort=top/n;
  printf ("Sonuc fonksiyonda yazdiriliyor...\n");
  printf ("A nin degerlerinin ortalamasi= %.2f",ort);
}
```

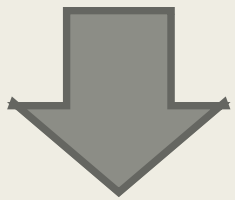
```
1. elemani giriniz:  45
2. elemani giriniz:  32
3. elemani giriniz:  56
4. elemani giriniz:  78
5. elemani giriniz:  90
Sonuc fonksiyonda yazdiriliyor...
A nin degerlerinin ortalamasi= 60.20
--------------------------------
```

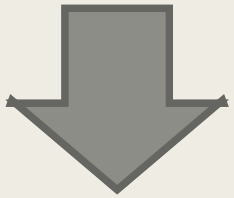
Örnek 2: Ana programdan 10 adet sayı giriliyor ve bu sayılar A isimli bir diziye aktarılıyor. A isimli dizideki elemanlar enbuyuk isimli bir fonksiyona gönderiliyor ve bu fonksiyon da dizinin elemanlarının en büyüğü bulunuyor. Bulunan en büyük değeri ana programda yazdıran bir C programı yazınız.

```
1. elemani giriniz: 34
2. elemani giriniz: 78
3. elemani giriniz: 90
4. elemani giriniz: 87
5. elemani giriniz: 65
6. elemani giriniz: 42
7. elemani giriniz: 20
8. elemani giriniz: 175
9. elemani giriniz: 228
10. elemani giriniz: 3
A nin en buyuk degeri=228
Sonuc ana fonksiyon da yazdirildi...

--------------------------------
```

C kodu:

```
#include <stdio.h>
#include <conio.h>
int enbuyuk(int B[], int n);

int main()
{ int A[10],enb,i;
  for (i=0;i<10;i++)
  {printf ("%d. elemani giriniz: ",i+1);
   scanf ("%d", &A[i]);
  }
  enb=enbuyuk(A,10);
  printf ("A nin en buyuk degeri=%d\n",enb);
  printf ("Sonuc ana fonksiyon da yazdirildi...\n");
  getch();
  return 0;
}
```





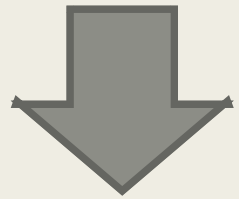
```
int enbuyuk(int B[], int n)
{int i,x;
 x=B[0];
 for (i=1;i<n;i++)
 {if (B[i]>x) x=B[i];
 }
  return x;
}
```

Örnek 3: Ana programdan 10 adet sayı giriliyor ve bu sayılar A isimli bir diziye aktarılıyor. A isimli dizideki elemanlar kbsirala isimli bir fonksiyona gönderiliyor ve bu fonksiyon da dizinin elemanları küçükten büyüğe sıralanıyor. A dizisinin elemanlarını ana programda, dizinin küçükten büyüğe sıralanmış halini kbsirala isimli fonksiyonda yazdıran bir C programı yazınız.

```
1. elemani giriniz: 44
2. elemani giriniz: 67
3. elemani giriniz: 89
4. elemani giriniz: 90
5. elemani giriniz: 99
6. elemani giriniz: 100
7. elemani giriniz: 43
8. elemani giriniz: 32
9. elemani giriniz: 35
10. elemani giriniz: 20
44  67  89  90  99  100  43  32  35  20
A dizisinin elemanlari ana fonksiyon da yazdirildi...

20  32  35  43  44  67  89  90  99  100
A dizisinin elemanlari sirali olarak kbsirala fonksiyonun da yazdirildi...

--------------------------------
```

C kodu:

```
#include <stdio.h>
#include <conio.h>
void kbsirala(int B[], int n);

int main()
{ int A[10],enb,i;
  for (i=0;i<10;i++)
  {printf ("%d. elemani giriniz: ",i+1);
   scanf ("%d", &A[i]);
  }
  for (i=0;i<10;i++)
  {printf ("%d  ",A[i]);}
  printf ("\n");
  printf ("A dizisinin elemanlari ana fonksiyon da
yazdirildi...\n");
  printf ("\n");
  kbsirala(A,10);
  getch();
  return 0;
}
```



```
void kbsirala(int B[], int n)
{int i,j,x;
 for (i=0;i<n-1;i++)
  {
   for (j=i+1;j<n;j++)
   {if (B[i]>B[j]) {x=B[i];
                    B[i]=B[j];
                                B[j]=x; }
   }
  }
for (i=0;i<n;i++)
  {printf ("%d  ",B[i]);}
  printf ("\n");
  printf ("A dizisinin elemanlari sirali olarak kbsirala
fonksiyonun da yazdirildi...\n");
}
```

Rekürsif (Özyinelemeli) Fonksiyonlar
(Kendi kendini çağıran Fonksiyonlar)

--- Kendi kendini çağıran fonksiyonlara Rekürsif fonksiyonlar denir.

--- Rekürsif fonksiyonlar en basit durumu ya da temel durumu nasıl çözeceğini bilir.

--- Rekürsif fonksiyon yeni bir kopyasını çağırarak nasıl yapacağını bilmediği problemi çözer.

--- Rekürsiflik işlemi bir önceki kopyaya bir sonuç aktarır ve fonksiyonun orijinal çağrısının en son sonucu döndürmesine kadar yukarıya devam eder.

--- Sonunda temel durum çözülmüş olur.

Örnek: Faktöriyel işlemini ele alalım.

--- 5! = 5 * 4 * 3 * 2 * 1

--- Biliyoruz ki

5! = 5 * 4!

4! = 4 * 3!

--- Böylece 5! = 5 *(4 * 3!) elde edilir.

--- 1! = 0! = 1 geri döndürülene kadar işlem devam edilir.

Yineleme şekilde hesaplanabilir:

--- 5! = 5 *(4 * 3!)

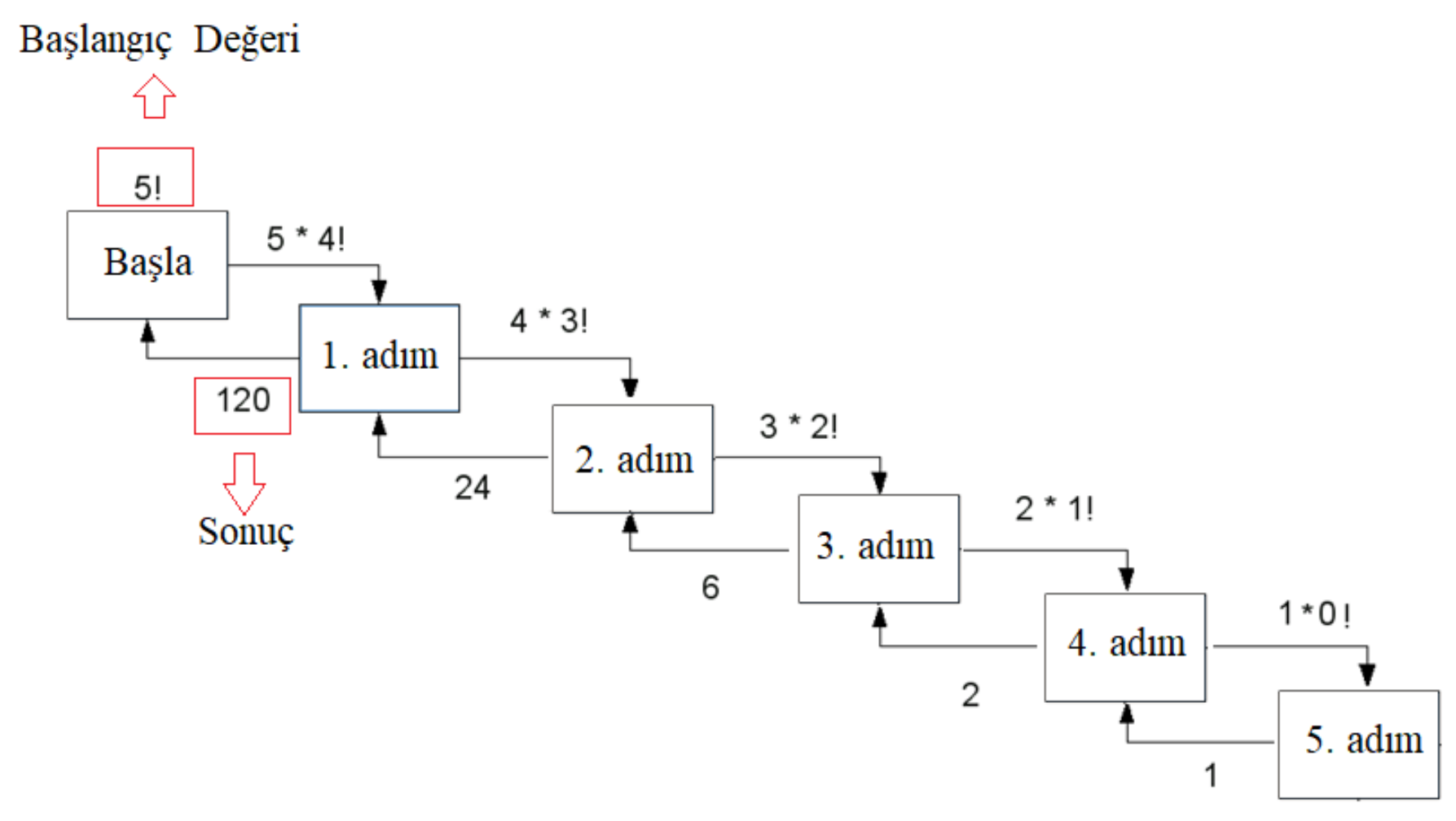
--- 5! = 5 *(4 * (3*2!))

--- 5! = 5 *(4 * (3*(2*1!)))

--- 5! = 5 *(4 * (3*(2*(1*0!))))

--- 5! = 5 *(4 * (3*(2*(1*1)))) = 120

Yineleme İşlemi:



Örnek 1: 1 den 10 a kadar olan sayıların faktöriyelini rekürsif olarak hesaplayan bir C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
long faktoriyel(long sayi);

int main()
{
   int i;
   for(i=1;i<=10;i++)
      printf("%3d!=%7ld\n",i,faktoriyel(i));
   getch();
   return 0;
}

long faktoriyel(long sayi)
{
    if(sayi<=1) return 1;
    else return(sayi*faktoriyel(sayi-1));
}
```

```
  1!=      1
  2!=      2
  3!=      6
  4!=     24
  5!=    120
  6!=    720
  7!=   5040
  8!=  40320
  9!= 362880
 10!=3628800


--------------------------------
```

Örnek 2: Kendisinden önce gelen iki tamsayının toplamı ile elde edilen sayı dizisine fibonacci sayı dizisi denir.

1 1 2 3 5 8 13 21 34 55…

Burada:

F[0]=1, F[1]=1, F[2]=2, F[3]=3, F[4]=5, F[5]=8, F[6]=13, F[7]=21… şeklindedir.

Klavyeden bir n tamsayısı giriliyor. n. adıma kadar olan Fibonacci dizisi elemanlarını ekrana yazdıran bir C programı yazınız.

Rekürsif olarak:

```
#include <stdio.h>
#include <conio.h>
int fibonacci(int n);

int main()
{   int sonuc,i,n;
    printf("Bir tamsayi giriniz:");
    scanf("%d",&n);
    for (i=0;i<=n;i++)
       {sonuc=fibonacci(i);
       printf("Fibonnaci(%d)=%d\n",i,sonuc);
       }
    getch();
    return 0;
}

int fibonacci(int n)
{ if(n==0 || n==1) return n;
    else return(fibonacci(n-1)+fibonacci(n-2));
}
```

```
Bir tamsayi giriniz:10
Fibonnaci(0)=0
Fibonnaci(1)=1
Fibonnaci(2)=1
Fibonnaci(3)=2
Fibonnaci(4)=3
Fibonnaci(5)=5
Fibonnaci(6)=8
Fibonnaci(7)=13
Fibonnaci(8)=21
Fibonnaci(9)=34
Fibonnaci(10)=55

--------------------------------
```

Dizi ile:

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

int main()
{   int F[50];
    int n,i;
    F[0]=0; F[1]=1;
    printf("Bir tamsayi giriniz:");
    scanf("%d",&n);
    for (i=2;i<=n;i++)
    F[i]=F[i-1]+F[i-2];

    for (i=0;i<=n;i++)
    printf("Fibonnaci(%d)=%d\n",i,F[i]);
    getch();

    return 0;
}
```

```
Bir tamsayi giriniz:10
Fibonnaci(0)=0
Fibonnaci(1)=1
Fibonnaci(2)=1
Fibonnaci(3)=2
Fibonnaci(4)=3
Fibonnaci(5)=5
Fibonnaci(6)=8
Fibonnaci(7)=13
Fibonnaci(8)=21
Fibonnaci(9)=34
Fibonnaci(10)=55

--------------------------------
```

--- Rekürsiflik, fibonacci sayısı hesaplanırken uygun değildir. n değeri büyüdükçe hesaplama süresi uzar.

--- Faktöriyel hesaplarken Rekürsiflik daha uygundur.

--- Ne zaman Rekürsiflik kullanacağımızı problemin tipine göre belirlemeliyiz…