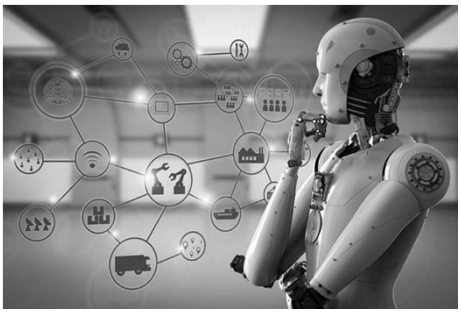
Bilgisayar ve Programlamaya Giriş



Hafta 13

İŞLEV (Fonksiyonlar(Functions))

Problemin çözümünde genellikle uzun programlara ihtiyaç duyulmaktadır. Binlerce satırdan oluşan bir programın yazılması ve anlaşılması zordur. Bu sebeple problemin daha kolay çözülebilen alt parçalarına (işlevlerine) ayrıştırılması gerekir.

İşlevler (fonksiyonlar / metotlar) bir kere yazılır ve program içerisinde bir ya da daha çok yerde çağrılarak kullanılır.

```
#include <xxxxx.h>

void fonk1();//fonksiyon bildirimi(function declaration-function protoytpe)

int main()
{
	..
	fonk1();   // fonksiyon cağırma (function call)
	..
	return 0;
}

void fonk1() // fonksiyon tanimlama (function definition)
{
	..
}
```

İşlev (Fonksiyon) örnekleri 1

```c
// toplama işlevi yapan bir fonksiyon yazalım
#include <stdio.h>
int topla(int a, int b); // fonksiyon bildirimi
                         // parametreler
int main(void)
{
  int a=1,b=-9;
  printf("a+b = %d\n",topla(a,b)); // fonksiyon cağrımı
                                   // argümanlar
  return 0;
}
// fonksiyon tanımı
int topla(int a, int b)
{
  return a+b;
}
```

```c
// gecikme işlevi yapan bir fonksiyon yazalım
#include<stdio.h>
void zaman(int n); //fonksiyon bildirimi/prototipi
//(function declaration-function protoytpe)

int main()
{
    printf("Program basi..Bekleyiniz..\n");
    zaman(50000); // fonksiyon cağrımı (function call)
    printf("Program sonu..\n\n");
    return 0;
}
void zaman(int n) // fonksiyon tanımı (function definition)
{
  int i,j;
  for(i=0;i<n;i++)
    for(j=0;j<n;j++);
}
```

İşlev (Fonksiyon) örnekleri 2

```c
//Farklı bildirimler ve çağrımlar
#include <stdio.h>

// fonksiyon bildirimleri
float topla1(float a, float b);
int topla2(int c, int d, int e);
void yazdir();

int main(void)
{
  float a=1.1,b=-9.0;
  int c=-21,d=1,e=2,sonuc;
   // fonksiyon cağrımları
  sonuc=topla2(c,d,e);
  printf("a+b = %5.2f\n",topla1(a,b));
  printf("c+d+e= %d\n\n",sonuc);
  yazdir();
  return 0;
}
```

```c
//devamı

// fonksiyon tanımları: topla1, topla2, yazdir
float topla1(float a, float b)
{
  return a+b;
}

int topla2(int c, int d, int e)
{
  return c+d+e;
}
void yazdir()
{
  printf("ana fonksiyon icinden cagrildim ve\n");
  printf("yazdir() isimli fonksiyon tarafindan yazdirildim\n\n");
}
```

İşlev (Fonksiyon) örnekleri 3

```c
/* Program, girilen sayının Çarpım tablosunu oluşturur */
#include <stdio.h>
void tablo(int no);
int main()
{
  int sayi;
  printf("\n Bir Sayi Giriniz : ");
  scanf("%d", &sayi);
  tablo(sayi);
  return 0;
}
void tablo(int no)
{
  int n = 0;
  printf("\n\n %d icin carpim tablosu\n ", no);
  for (n = 1; n <= 10; n++)
      printf("\n %d * %d = %d", no, n, (no*n));
}
```

İşlev (Fonksiyon) örnekleri 4 (dizileri çarpmıştık)

```c
// geçen hafta 3x3 Matris çarpımı yapmıştık
/* Bu hafta fonksiyonlar ile 3x3 Matris çarpımı*/
#include<stdio.h>
//Fonksiyon bildirimleri
void dizioku();
void dizicarp();
void diziyaz();

int i,j,k,toplam;
int A[3][3],B[3][3],C[3][3];

int main()
{
  dizioku();
  dizicarp();
  diziyaz();
  return 0;
}
```

```c
//devamı
//Fonksiyon Tanımları
void dizioku()
{
  printf("A dizisinin elemanlarini giriniz : ");
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      scanf("%d",&A[i][j]);

  printf("B dizisinin elemanlarini giriniz : ");
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      scanf("%d",&B[i][j]);
}
```

```c
//devamı

void dizicarp()
{
  for(i=0;i<3;i++)

    for(j=0;j<3;j++)
    {
      C[i][j]=0;
      for(k=0;k<3;k++)
      {
        C[i][j]+=A[i][k]*B[k][j];
      }
    }
}

void diziyaz()
{
  for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
      printf("%3d",C[i][j]);
    printf("\n");
  }
}
```

İşlev (Fonksiyon) örnekleri 5

```c
// Derece Radyan Dönüşümü
#include <stdio.h>
void menu(void);
int main()
{
  int secim;
  float aci;

  menu();
  scanf("%d", &secim);
  while (secim != 3)
  {
    switch (secim)
    {
    case 1:
      printf("Radyan cinsinden açı => ");
      scanf("%f", &aci);
      printf("derece cinsinden açı %6.2f \n", aci / 3.141 * 180);
      break;
    case 2:
      printf("Derece cinsinden açı => ");
      scanf("%f", &aci);
      printf("radyan cinsinden açı %6.2f \n", aci / 180 * 3.141);
      break;
    }
    menu();
    scanf("%d", &secim);
  }
  return 0;
}
```

```c
/* devamı*/

void menu(void)
{
  printf("\n** Açı Radyan Dönüşümü **\n");
  printf(" 1. radyan => derece\n");
  printf(" 2. derece => radyan\n");
  printf(" 3. bitir\n");
  printf("Seçim ==> ");
}
```

İşlev (Fonksiyon) örnekleri 6

```c
//diziyi fonksiyona yollama işlemi

#include <stdio.h>


int fonk(int[],int);

int main()
{
  int sonuc;
  int dizi[5]={3,4,7,1,9};
  sonuc=fonk(dizi,5);
  printf("%d",sonuc);
  return 0;
}
```

```c
//devamı


int fonk(int d[],int s)
{
  int t=0;
  for(int i=0;i<s;i++)
  {
     t+=d[i];
  }
  return t;
}
```

İşlev (Fonksiyon) örnekleri 7 (Faaliyet Alanı)

```c
/* Değişkenlerin kapsamları 1 */
#include <stdio.h>

void fonksiyon1();

int a;

int main()
{
  int a = 4; //int?

  printf("%d\n",a);

  fonksiyon1();

  printf("%d\n",a);

  return 0;
}

void fonksiyon1()
{
  a = 10;
  printf("%d\n",a);
}
```

```c
/* Değişkenlerin kapsamları 2 */
/* b değişkeni main() fonksiyonundan sonra tanımlandığı için main
fonksiyonunun içinde geçerli değildir */

#include <stdio.h>

void yaz(); // Fonksiyon tanımlanıyor
int a = 4; // Global değişken tanımlanıyor

int main()
{
  printf("%d\n",a);
  yaz();
  //printf("%d\n",b); //HATA!
  return 0;
}

int b = 5; // Global değişken tanımlanıyor

void yaz()
{
  printf("%d\n",b);
}
```

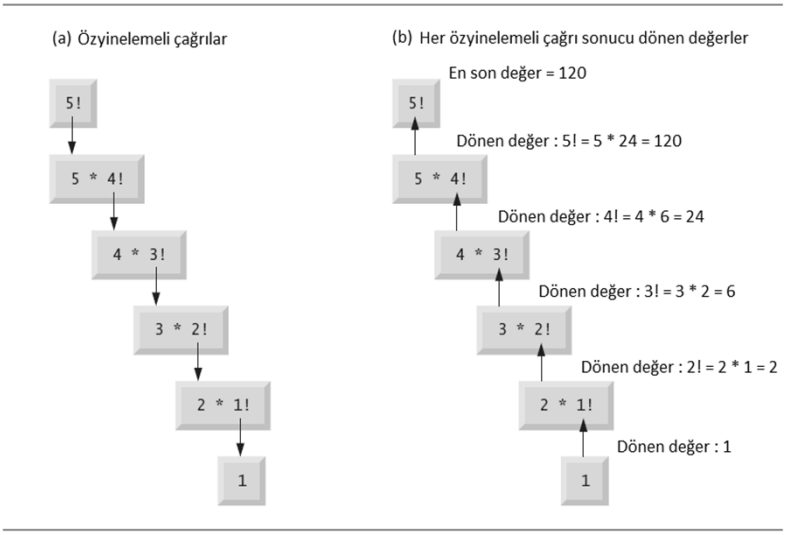
Özyineleme (Recursion)

Doğrudan veya dolaylı olarak kendisini çağıran fonksiyonlara **özyineli fonksiyonlar** (recursive functions) denir.

Özyineli fonksiyonlar sadece **en basit durum**u (simplest case) çözmeyi bilirler. Bu duruma **kesin durum** (base-case) da denir. Yani bir problem özyineli olmayan basit bir çözüme sahiptir ve özyineli fonksiyonlar bu durumu kullanırlar. Bir fonksiyon kesin durum ile çağırılırsa sadece basit bir şekilde değer döndürür. Fonksiyon çok daha karmaşık bir problemi çözüyorsa o problemi kavramsal iki parçaya böler. Bunlardan ilki fonksiyonun nasıl yapacağını bildiği, ikincisi ise nasıl yapacağını bilmediğidir. Özyinelemeyi yapabilmek için ikinci kısım orjinal probleme basit bir şekilde benzemek zorundadır. Bu küçük problem üzerinde çalışması için, fonksiyon kendisinin bir kopyasını çağırır. Bu işleme özyineli çağrı (recursive call) ya da özyineleme adımı (recursion step) denir. Özyineleme adımı return anahtar kelimesini içerir, çünkü buradan dönecek olan sonuç problemin bilinen kısmına eklenecektir.

Özyineleme – Faktöriyel Hesabı

5!'in özyineli hesabı



Özyineleme – Faktöriyel Hesabı

```
//faktöriyelin normal ve özyinelemeli bulunuşu :
#include <stdio.h>

int faktoriyel(int); //normal
int faktoriyelR(int); //recursive-özyineleme

int main()
{
  int sayi;
  printf("Bir sayi giriniz: \n");
  scanf("%d", &sayi);

  printf("%d\n", faktoriyel(sayi)); //normal
  printf("%d\n", faktoriyelR(sayi)); //recursive-
özyineleme

  return 0;
}
```

```
//devamı:

int faktoriyel(int sayi)
{//normal
  int sonuc = 1;
  for (int i = sayi; i>=1; i--)
    sonuc *= i;
  return sonuc;
}

int faktoriyelR(int sayi)
{//recursive-özyineleme
  if (sayi <= 1)
    return 1;
  else
    return sayi*faktoriyelR(sayi - 1);
}
```

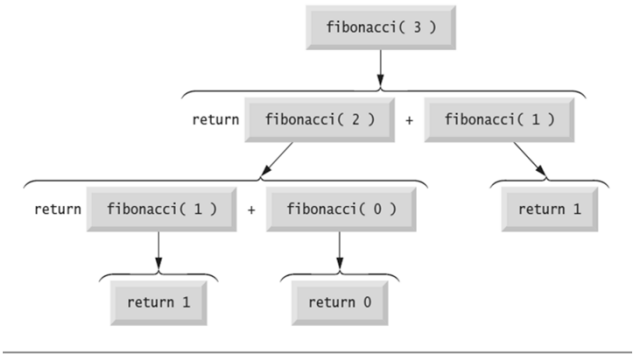
Özyineleme – Fibonacci hesabı

Fibonacci serisinin özyineli hesabı

fibonacci(0) = 0
fibonacci(1) = 1
fibonacci(*n*) = fibonacci(*n-1*) + fibonacci(*n-2*)

0 1 1 2 3 5 8 13 21 34 55 89 …

Fibonacci(3) değerini bulalım:



Özyineleme – Fibonacci Hesabı

```
//fibonacci serisinin normal ve özyinelemeli bulunuşu
#include <stdio.h>

int fibonacci(int);
int fibonacciR(int);

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

  printf("f(%d) : %d\n", sayi, fibonacci(sayi));
  printf("fr(%d) : %d\n", sayi, fibonacciR(sayi));
  return 0;
}
```

```
//devamı:
int fibonacci(int sayi)
{
  int sayi1 = 0;
  int sayi2 = 1;
  int sonuc = 0;
  for (int i = 2; i <= sayi; i++)
  {
    sonuc = sayi1 + sayi2;
    sayi1 = sayi2;
    sayi2 = sonuc;
  }
  return sonuc;
}


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