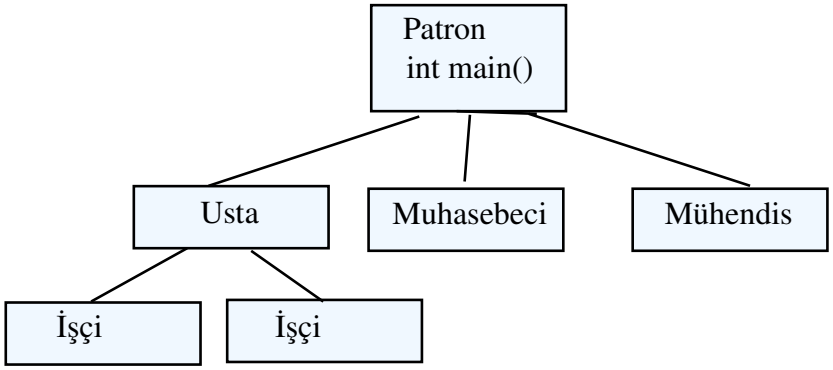
# FONKSİYONLARA GİRİŞ

Fonksiyonlar, programları ufak parçalara ayırıp kolaylıkla yönetebilmemize yarayan kod parçalarıdır. Eğer biz bir işi tekrar tekrar yapıyorsak ve bu işi yapan kod çok uzunsa; ana programı sadeleştirmek, daha anlaşılır yapmak için fonksiyon kullanma ihtiyacı duyarız.




Yukarıdaki şekil programların parçalanmasına bir örnektir. main() fonksiyonumuz programın patronudur ve programda ne yapılacağını bilir, ona göre kendi altında bulunan Usta, Muhasebeci ve Mühendis'e emirler vererek yapılması gerekenleri yaptırır. Bu kişiler kendi işlerini yaparken bir yandan da kendi altlarında İşçiler varsa, onlara da bir takım işler yaptırırlar. Bu iş bölümü sayesinde herkes kendi alanında yoğunlaştığı, kimin ne yaptığı belli olduğu için tam verim sağlanır. Görüldüğü üzere tüm bu kişilerin patronun eşgüdümünde yaptığı çalışma sonucunda ise program oluşur.

Diyelim ki biz bu programın ne yaptığını öğrenmek istiyoruz. O halde, gidip de işçiye veya muhasebeciye sormak mı daha mantıklıdır, yoksa patrona sormak mı? Tabii ki patrona sormak. Çünkü o, tüm yapılanları bilmekte ve bizzat yönetmektedir. O halde mühendise gidip projenin teknik detaylarını, muhasebeciye gidip iktisadi detaylarını öğrenmektense patrondan neler

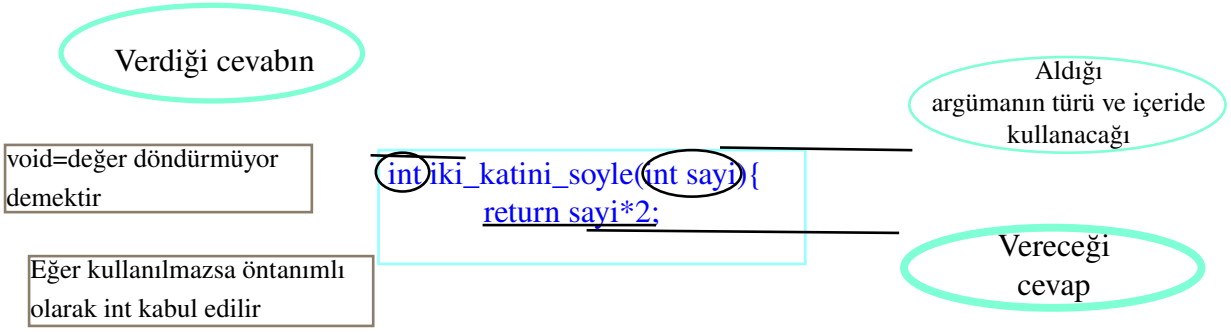
- Sistem ve malzeme için ne kadar paraya ihtiyacımız olduğunu mühendise sordum
- Ne kadar işçiye ihtiyacımız olduğunu ustaya sordum.
- Bu iki bilgiyi muhasebeciye söyledim ve ondan bana paramızın yetip yetmediğini tespit etmesini istedim.
- Muhasebeci paramızın yettiğini söyledi, bundan dolayı usta ve mühendise inşaata başlama emrini verdim.
- İnşaat süresince girdişat hakkında usta, muhasebeci ve mühendisten rapor istedim. Raporları değerlendirip projenin devamına karar verdim.
- Usta inşaatın bittiğini bildirdi.
- Emlakçıya binayı gezdirdim.
- Müşteri çıktı ve

Bunlar bizim işimize yarayabilecek bilgilerdi. Tutup da işçiye "boya yapmak mı daha zordu, demir döşemek mi?" diye sormak bize bir şey kazandırmaz. Hatta herkes bu tür detayları sorduğumuzda kafamızın nasıl karıştığını düşünün.

Dikkat ettiyseniz patronun rapor aldığını söyledik. Raporlama, programlamada karar-kontrol mekanizmalarında ve eşgüdümlü çalışma yürütmede çok önemlidir. Çünkü bir fonksiyonun çalışması, diğer bir fonskiyonun işinin sonucuna bağlıdır.

Fonksiyonlar arasındaki bu iletişim, argüman(parametre) ve döndürmelerle olur. Bir fonksiyonun aldığı şeyler (argüman) vardır, bir de döndürdüğü cevap vardır.

| | Değer Döndüren | Değer Döndürmeyen |
|---|---|---|
| Argüman Alan | int iki_katini_soyle(int sayi){<br>return sayi*2;<br>} | void sayiyi_yaz(int sayi){<br>printf("sayi: %d",sayi);<br>} |
| Argüman Almayan | int sekiz_cevabi_ver(){<br>return 8;<br>} | void merhaba_de(){<br>printf("Merhaba!");<br>} |




```
#include <stdio.h>

int iki_katini_soyle(int);

int main(){
	int x, cevap;

	printf("Bir sayi giriniz : ");
	scanf("%d",&x);

	cevap=iki_katini_soyle(x);
	printf("Girdiğiniz sayı olan %d sayisinin iki kati %d olarak hesaplanmıştır.", x, cevap);

	return 0;
}

int iki_katini_soyle(int sayi){
	return sayi*2;
```

Yukarıda, bir fonksiyon kullanım örneği verilmiştir. Bu örneğe göre kullanıcıdan bir sayı isteniyor, bu sayı x değişkenine atanıyor. Daha sonra bu x sayısını iki_katini_soyle fonksiyonuna argüman olarak gönderiyoruz(parantez içine x yazarak). Bunu yaptığımız zaman bilgisayar x sayısının değerine bakıyor, diyelim ki 5. Bunu değeri, iki_katini_soyle(int sayi) kısmında "sayi"ya kopyalıyor. Yani fonksiyonun içine geçildiği zaman sayi değişkeninin değeri artık 5. Fonksiyon, "sayi"nın yani, 5'in ikiyle çarpılmış halini return ile döndürüyor. Yani 10 cevabını veriyor. Bu 10 cevabı nereye gidiyor sizce?

cevap=iki_katini_soyle(x); ======> cevap=10;
haline geliyor. Yani, "iki_katini_soyle(x)" ifadesini silip yerine 10 yazmış gibi oluyoruz.

O halde, cevap değişkeninin değeri artık 10.
printf ile bastırdığımızda ise,
"Girdiğiniz sayı olan 5 sayisinin iki kati 10 olarak hesaplanmıştır."
yazısı basılır ekrana.

cevap=sekiz_de(); dersek ise cevap=8; olmuş olur.(Tablodaki fonksiyona göre)
merhaba_de(); dersek
"Merhaba!" yazılacaktır ekrana.

Burada dikkat edilmesi gereken nokta şu:
cevap=merhaba_de(); diyemezsiniz. Çünkü merhaba_de fonksiyonunun türü void , yani değer döndürmüyor.

Dikkat ettiyseniz, yukarıdaki program kodunda int main() in üzerinde fonksiyon türünü, adını, argüman türünü ve argümanı yazmışız. Buna da prototip denir. :

int iki_katini_soyle(int);

Bu demektir ki, "Derleyici, bak seni baştan uyarıyorum, ileride karşına iki_katini_soyle diye bir fonksiyon çıkabilir, bu fonksiyon bir tane integer(tam sayı) argüman alıp sana integer cevap döndürecek. Ona göre davran(bellekte yer ayır vs)."

```
#include <stdio.h>

int carp(int, int);

int main(){
	int x, cevap;

	printf("iki sayi giriniz : ");
	scanf("%d %d",&x,&y);

	cevap=carp(x,y);
	printf("%d x %d = %d olarak hesaplanmıştır.", x, y, cevap);

	return 0;
}

int carp(int sayi1, int sayi2){
	return sayi1*sayi2;
}
```

Burada ise gördüğünüz üzere carp fonksiyonu iki tane integer argüman alıyor, bunların çarpımını döndürüyor. Bu durumda argümanları virgül ile ayırıyoruz ve her argümanın türünü ayrı ayrı belirtiyoruz. Dikkat edilecek bir diğer nokta ise, yazdığımız fonksiyonlar main fonksiyonunun içerisinde değiller! Hepsi birbirinden bağımsız bireyler ve iç işlerinde bağımsız, dış işlerinde main'e bağlı ülkeler gibiler.

Yani, bir fonksiyonun kendi içerisinde kullandığı değişkenler sadece fonksiyon çalıştığı sürece geçerlidir. Fonksiyonun çalışması sona erdiği anda içindeki değişkenler yok olur. Geriye sadece ekrana yazı yazdırmak, bir değer döndürmek gibi etkileri kalır ki, döndürülen değer başka bir fonksiyon tarafından havada kapılmış ve kullanılmıştır bile.
Şimdi de, kullanıcının girdiği 3 sayıdan en büyüğünü fonksiyon kullanarak hesaplayan bir program yazalım:

```
#include <stdio.h>
int enbuyuk(int,int,int);

int main(){

	int s1,s2,s3;

	printf("Uc sayi giriniz: ");
	scanf("%d%d%d",&s1,&s2,&s3);

	printf("En buyugu : %d",enbuyuk(s1,s2,s3) );

	return 0;
}

int enbuyuk(int x, int y, int z){
	int enbyk;
	enbyk=x; //Burada dikkat edin, değişken ismi ile fonksiyon ismi aynı olmamalı!
	if (y>enbyk)
		enbyk=y;
	if(z>enbyk)
		enbyk=z;

	return enbyk;
}
```

# Rastgele Sayı Üretimi

Rastgele sayı üretmek için bazı hazır fonksiyonları kullanırız. Bunlar, srand ve rand fonksiyonlarıdır. srand fonksiyonu, seed rand 'dan gelir. Yani rastgele sayının neye dayanarak üretileceğini bu fonksiyonla belirleriz. Eğer bu sayı sabit bir sayı olursa, programı her çalıştırdığımızda "rastgele" sayı hep aynı çıkar. Bu yüzden sürekli değişen bir kaynakla beslemeliyiz. Bu ise zamandır. Zamanı kullanabilmek için time.h, random fonksiyonlarını kullanabilmek için ise stdlib.h kütüphanelerini yüklememiz gerekir. Hemen örnek verelim:

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

int main(){
	int rastgele_sayi;
	srand(time(NULL)); //Kalıptır,rand() fonksiyonunu kullanmadan önce çağırmak gerekir.

	rastgele_sayi=rand()%100;
	printf("100'den kucuk rastgele sayi %d olarak belirlendi.", rastgele_sayi);

	return 0;
}
```

Burada rand() fonksiyonu rastgele bir sayı üretiyor. Örneğin 3984. Sonra, bunun 100'e bölümünden kalanı alıyoruz: 84. Böylece 100'den küçük rastgele bir sayı elde edilmiş oldu. Bu yöntem ile [0-99] kapalı aralığı arasında rastgele sayı üretilebilir. Eğer 1 ile 100 arasında olmasını isteseydik:

```
	rastgele_sayi=1+rand()%100;
```

derdik. Eğer 2 ile 96 arasında olmasını isteseydik: (96-2)+1=95 (saymanın temel kuralı)

```
	rastgele_sayi=2+rand()%95;
```

derdik.

## SCOPE VE STATIK