1) C PROGRAMLAMAYA GİRİŞ

Programlama: Bilgisayarın donanıma nasıl davranacağını anlatan, bilgisayara yön veren komutlar, kelimeler, aritmetik işlemlerdir.

Bir C Kodu Nasıl Çalışır

Bir c kodu çalışmadan önce 4 aşamadan geçer bu aşamalar sırasıla:Edit, preprocessor, compiler ve linker aşamalarıdır.

1)Edit:
Kod “yazmak” derken aslında bu aşamadan bahsediyoruz. Bu aşama bizim belirli kurallar dahilinde en özgür olduğumuz aşamadır.

2)Preprocessor (önişlemci):
İsimdeki işlemci kelimesi sizi yanıltmasın, preprocessor bir sonraki aşama olan compiler ile birlikte gelen bir yazılımdır.Preprocessor’ın görevlerini vakti geldikçe anlatacağız. Şimdilik bilmemiz gereken şey: Başında ‘#’(diyez-sharp) işareti olan kod satırı bir önişlemci komutudur.

3)Compiler (derleyici)
Derleme işlemi bir kodun 0 ve 1’lerden oluşan makine koduna çevrilmesidir.

4)Linker (bağlayıcı)
Compiler ne yazık ki her kod parçasını tanıyamaz bu yüzden onları makine koduna çeviremez ve boşluklar bırakır. Bu boşluklar daha önceden 0 1’e çevrilmiş kodlar tarafından linker aracılığıyla doldurulur.

1.2 Standart Nedir

Standart kelimesini “sürüm” gibi düşünebiliriz. C proglamlama dilinin 3 standardı vardır:
c11 - 2011
c99 - 1999
c89 – 1989

Biz kursumuzda c99 standardını kullanacağız. Bu farkı bilmemiz önemli çünkü c89 standardında yazdığımız kod c99 standardında çalışmayabilir. 89 ile 99 standardı arasındaki farkları sırası geldikçe öğreneceğiz.

1.3 Kodlarımızı nerede yazacağız

Ben visual studio kullanıyorum.
Windows kullanıcıları için kurulum ve kullanım kolaylığı açısından Dev-c++ tavsiye ediyorum

İlk Kod

Şimdi Dev-c++ ta “Dosya” yazan yerin altındaki dikdörtgene tıklayıp yeni bir belge oluşturalım ve ctrl + s ile herhangi bir yere ilk_kod.c ismi ile kaydedelim.

İşte ilk kodumuz

```c
#include <stdio.h>

int main(){

    printf("Hello World!\n");
    //burada bir fonksiyonumuz var

    return 0;
}
```

Gelin ilk kodumuzu inceleyelim:

#include <stdio.h>
başında # işareti olduğuna göre bu bir önişlemci komutudur

std: standart
i: input-girdi
o: output-çıktı
h:header file kısaltmasıdır.

Çok basit düşünerek diyebiliriz ki: standart input yani bilgisayara en basit yoldan veri girebildiğimiz yer, klavyedir. Standart input ise monitördür. Sonuç olarak diyebiliriz ki bu kod satırı ile girdi ve çıktı işlemlerini yapabiliyoruz.

int main()
bu yapıyı fonksiyonlar kısmına gelince çok daha iyi anlayacağız. Şimdilik şöyle diyelim. main isminde bir fonksiyonumuz var ve bu bizim programımız. Yani “C programı” dediğimizde aslında bir main fonksiyonundan bahsediyoruz.

Başındaki int ifadesi ise fonksiyon bittiğinde değerinin bir tam sayı (integer) olacağını gösteriyor.

printf(“Hello World!\n”);
printf, ekrana yazı yazdırmamızı sağlayan fonksiyondur. Bu fonksiyonu kullanmak için ilk satırdaki include işlemini yapmalıyız.

Fonksiyonun içine, yazacağımız şeyleri çift tırnak - double quote – arasına koyarak yazıyoruz.

Kodu çalıştırdığımızda \n ifadesinin ekrana yazılmadığını göreceğiz çünkü \n özel bir ifadedir. Alt satıra geçmeye yarar.Özel bir karakterdir. Bir kaçış dizisidir (escape sequence)

Kod satırlarımızın bittiğini belirtmek için sonuna ; (semicolon) koyarız

//burada bir fonksiyonumuz var
bu yazının başında iki tane bölme işareti var (//). bu ifade ile başlayan satıra “yorum satırı” (comment line) diyoruz. Yorum satırları önemlidir çünkü bazen kendimiz bazen de kodu okuyacak başka insanların kodu anlaması için kodumuza tabiri caizse “not bırakmamız” gerekebilir. Kod satırları önişlemci tarafından kaldırılır böylece compiler hata vermez

// ifadesi sadece olduğu satırı yorum satırına çevirir. Eğer birden fazla yorum satırı yazacaksak // ifadesini alt alta kullanmamız gerekebilir. Mesela:

```
//coklu
//yorum satirlari
//boyle yazılıyor
```

çoklu yorum satırları için /* */ ifadesini de kullanabiliriz

```
/*
coklu yorum satirlari
boyle de yazilabilir
*/
```

return 0;
int main ifadesinin işi bitince değerinin bir tamsayı olacağını söylemiştik. O değerin sıfır (0) olduğunu bu satırdan anlıyoruz

{}
printf gibi fonksiyonlarımızı veya başka kod satırlarımızı süslü parantezler arasına yazarız. Süslü parantezlerin arasındaki kodlara “blok” denir.

İlk Kodumuzu Çalıştıralım

windows

Kodumuzu kaydettikten sonra Dev-c++ta

 bu kutuların olduğu yere gidip sağdan 3. ye basıp çalıştıralım bu yeterli olacaktır.

Linux-visual studio

linux kullanıcılar için durum biraz daha karmaşık. İsterseniz uç birimden cd komutlarıyla kodu kaydettiğimiz dosyaya gidelim ve aşağıdaki komutu yazalım

gcc ilk_kod.c -o ilk_kod

bu komutu açıklayalım:
gcc: compiler'ın ismi
ilk_kod.c: derleyeceğimiz c kodu
-o: derle ve bağla (compile and link)
ilk_kod: kodumuzun çalışmaya hazır halinin ismi, ilk_kod yazmak zorunda değiliz istersek benim_kodum gibi bir isim de seçebilirdik. Önemli olan bir nokta var ki: iki kelimelik dosya isimlerini boşluk ile yazmamalıyız o yüzden iki kelime arasına "_" koyuyoruz.

Kod derlendi ve bağlandı. Şimdi terminale

./ilk_kod

komutunu giriyoruz ve çıktımızı alıyoruz.

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$ gcc ilk_kod.c -o ilk_kod
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$ ./ilk_kod
Hello World!
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$
```

Şimdi printf içindeki \n ifadesini silip kodu kaydedelim. Tekrar derleyip çalıştıralım

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$ gcc ilk_kod.c -o ilk_kod
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$ ./ilk_kod
Hello World!mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM$
```

Farkettiyseniz Hello World! İfadesinden sonra aşağıya geçilmemiş. İşte \n ifadesi bu işe yarıyor.

Değişken Tanımlamak (definition)

Değişken tanımlamak hafızada bir alan ayırmak, o alana isim vermek ve opsiyonel olarak o alanda veri depolamaktır.

Bir değişken nasıl tanımlanır?

veri_tipi değişken_ismi = başlangıç_değeri(opsiyonel) ;

İngilizce olarak:

data_type identifier = initial_value ;

veri tipi nedir?

Verilerin farklı türleri vardır mesala bir sayı ,tam sayı veya ondalık sayı ya da bir harf de hafızada tutulabilir. Bu değerlerin tutulması için hafızada farklı miktarda alan ayrılır ve ne kadar alan ayırılacağını C bilmek ister. Bundan dolayı değişkenlerimizi veri tipleriyle birlikte tanımlarız

TAMSAYI değerlerini tutmak için bir çok veri tipi vardır. 2 tanesi yeterli olacaktır.

Veri tipi ismi	Alabileceği değer aralığı	Kapladığı alan (byte)
int	–2,147,483,648 ile 2,147,483,647 arası (yaklaşık 2 milyar)	4
long int	–9,223,372,036,854,775,808 ile 9,223,372,036,854,775,807 arası	8

Bu aralıklar ve kaplanılan alan işletim sistemine göre değişebilir. Fakat biz bu sınırlara ulaşmayan değerleri kullanacağız.

Mesela i isminde değeri 5 olan bir değişken tanımlayalım

int i = 5; (noktalı virgülü unutmayalım)

ONDALIK SAYI değerleri için 2 tane veri tipi kullanacağız.

Veri tipi ismi	Alabileceği değer aralığı	Kapladığı alan (byte)
float	1.2E-38 ile 3.4E+38 arası (E38 = 10^38)	4
double	2.3E-308 ile 1.7E+308 arası	8

Mesela değeri 3.14 olan pi isminde değişkenleri tanımlayalım
float pi = 3.14;
double pi = 3.14;

KARAKTER değerlerini (harfler ve % & ! gibi özel karakterler) tutmak için char veri tipini kullanıyoruz

Veri tipi	Değer aralığı	Kapladığı alan (byte)
char	Tüm ingilizce klavye karakterleri. (büyük küçük harf farketmez) Türkçe harfler için bir takım düzenlemeler yapmamız gerekiyor	1

NOT = Bir değişken 2 defa tanımlanamaz

NOT:
char tipinden değişken tanımlarken girilen karakteri tek tırnak ‘ ’ içinde yazarız

Mesela değeri a olan bir değişken tanımlayalım

char harf = ‘a’;

NOT:
başlangıç değeri vermek zorunda değiliz

int sayi;
şeklinde bir tanımlama yapabiliriz ama bu durumda sayi değişkeninin değeri rastgele bir değer olacaktır

NOT:
aynı satırda, aynı veri tipinden birden fazla değişken tanımlayabiliriz. Bu durumda değişkenler arasına virgül, satırın sonuna noktalı virgül koyarız.

```
int sayi1, sayi2 = 6, sayi3;
float sayi4, sayi5, sayi6 ;
```

Değişken Tanımlama Kuralları

Değişken tanımlarken bazı kurallarımız var

1) Değişken isimleri Türkçe karakter içeremez
   int sayı = 0; yerine
   int sayi = 0; yazılır

2)Değişken isimleri sayı ile başlayamaz ama sayı içerebilir
   int 1sayi = 5; yerine
   int sayi1 = 5; yazılır

3)Değişken isimleri alt tire “_” hariç bir özel karakter (. , % # ^) ve boşluk içeremez. Değişken isimleri alt tire ile başlayabilir ama alt tire genellikle iki kelimelik değişkenler tanımlamak için kullanılır

double pi sayisi = 3.14; geçersiz tanımlama
double pi_sayisi = 3.14; uygun tanımlama

int sayi? = 5; geçersiz
int sayi = 5; uygun

4)C bir “case sensitive” dildir yani büyük harf - küçük harf ayrımı yapar.
int Sayi = 55;
int sayi = 55; Bunlar farklı değişkenlerdir.

5)Bazı kelimeler C dilinde önceden tanımlıdır. Bunlara “keyword / anahtar kelime” denir ve değişken isimleri bu kelimeler olamaz.
Bu kelimeler:

```
auto     double int      struct
break    else   long     switch
case     enum   register typedef
char     extern return   union
continue for    signed   void
do       if     static   while
default  goto   sizeof   volatile
const    float  short    unsigned
```

6) Değişken isimleri anlaşılabilir olmalıdır mesela
float mb = 1.75; yerine
float mahmut_boy = 1.75; şeklindeki bir tanımlama hem bizim “mb” isimli değişkenin açılımı aklımızda tutmamız gerekmez hem de daha okunulabilir kod yazmış oluruz.

NOT: eğer int i = 3.14 gibi bir tanımlama yaparsak ne olacağını type casting kısmında göreceğiz şimdilik bunlara takılmayın

Atama (assignment)

```c
#include <stdio.h>

int main(){

    //tanımlama
    int sayi = 5;

    //atama
    sayi = 5;
    //isterseniz aynı değeri bile atayabilirsiniz

    sayi = 10;
    //isterseniz de farklı bir değer atayabilirsiniz

    return 0;
}
```

Atama yapmak daha önceden tanımlanmış bir değişkene tekrardan değer vermektir.

Değişkenleri yazdırma

Yazdırmak için printf fonksiyonunu kullanacağız fakat demiştik ki yazacağımız şeyleri “” arasına koymalıyız. Değişkenlerimiz “” arasındaki değerler olmadığı için değişkenleri printf fonksiyonunun içine farklı bir şekilde koymamız gerek.

String biçimlendirerek (formatted string)

Diyelim ki int veri tipinden bir değişken yazdıracağız bunun için “%d” ifadesini kullanırız. Çift tırnaklı kısmından sonra virgül koyarak yazdırmamak istediğimiz değişkeni yazıyoruz. (Virgül operatörü soldan sağa çalışır)

```
int sayi = 5;
printf("%d\n",sayi);
```

Çıktı:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ ./ders
5
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$
```

Her değişken için farklı bir sembol vardır. Bu sembollere “format specifier / dönüşüm belirteci” denir.

Veri tipi	Format specifier
int	%d
long int	%ld
float	%f
double	%f
char	%c

İstersek bir printf içerisinde birden fazla değişken basabiliriz. Burada önemli olan şey değişkenlerin virgülden sonraki sıralamalarıyla bastırılmasıdır.

```
#include <stdio.h>

int main(){

   int ilk_sayi = 5;
   int ikinci_sayi = 10;

   printf("%d %d",ilk_sayi, ikinci_sayi);

    return 0;
}
```

Başka bir örnek:

```
double boy = 1.75;
int kilo = 80;
char cinsiyet = 'E';

printf("%f %d %c\n",boy, kilo, cinsiyet);
```

Çıktı:
1.750000 80 E

Varsayılan olarak double veri tipi virgülden sonra 6 sayı ile birlikte basılıyor. Bu yüzden 1.75 yerine 1.750000 basıldı.

SCANF FONKSİYONU

Şu ana kadar değişkenlerimizin değerlerini hep biz belirledik ama birçok zaman kullanıcıdan veri alıp bu verileri değişkenlerde depolamamız gerekecek. Kullanıcıdan veri almak için scanf fonksiyonunu kullanırız.

scanf fonksiyonunu kullanmak için
#include <stdio.h>

Diyelim ki bir kullanıcının yaşını alıp ekrana basan bir program yazmak istiyoruz:

İlk önce yaş değişkenini depolayacak değişkeni tanımlamalıyız. Yaş, bir ondalık sayı veya bir harf değildir. Bir tam sayıdır . Bu yüzden int değişken tanımlayacağız.

```
int yas;
```

bir başlangıç değeri verebilirdik ama zaten değişeceği için vermedik.

```
printf("lutfen yasinizi giriniz: ");
```

printf fonksiyonu ile kullanıcı harekete geçmeye teşvik ediyoruz ve ne yazması gerektiğini söylüyoruz. \n işareti koymadım çünkü kullanıcının yazdığım mesaj ile aynı satırda işlem yapmasını istiyorum.

```
scanf("%d",&yas);
```

Scanf fonksiyonu içinde \n kullanmıyoruz.

%d ifadesi kullanıcıdan bir tam sayı (integer) değişkeni alacağımı compiler'a belirtiyor.

& (ampersand) ifadesi kendisinden sonra gelen değişkenin adresini belirtiyor.

Yani scanf fonksiyonu kullanıcıdan bir değer alıp onu yas değişkeni için ayırdığı alana koyuyor. Artık yas değişkeninin değeri kullanıcının klavyeden (klavye = standart input) girdiği sayı.

Sıra yas değişkenini bastırmakta

```
printf("yasiniz: %d\n",yas);
```

```c
#include <stdio.h>

int main(){

    int yas;
    printf("lutfen yasinizi giriniz: ");
    scanf("%d",&yas);
    printf("yasiniz: %d\n",yas);

    return 0;
}
```

Tam program

gördüğünüz gibi scanf fonksiyonu içinde %d kullandık yani değişkenin veri tipine göre farklı format specifier'lar kullanacağız.

Veri tipi	Format specifier
int	%d
long int	%ld
float	%f
double	%lf
char	%c

bu tablonun printf kısmındaki tablodan tek farkı double tipinin %lf ile alınması.

Kullanıcıdan 2 tane sayı alan programı yazalım:

```c
#include <stdio.h>

int main(){

    int ilk_sayi;
    double ikinci_sayi;
    printf("sayilar: ");
    scanf("%d %lf",&ilk_sayi, &ikinci_sayi);

    return 0;
}
```

istersek "sayı (boşluk) sayı (enter)" olarak istersek de "sayı (enter) sayı (enter)" şeklinde sayılarımızı girebiliriz.

ARİTMETİK

C dili sayılar ve değişkenlerle matematiksel işlemler yapmamıza izin verir.
İşlemin sembolüne operatör, işlem yapılan değerlere operant denir.

TOPLAMA

+ işareti kullanılır.

```c
#include <stdio.h>

int main(){

    //bir değişkeni iki sayının toplamı olarak yazabiliriz
    int sayi1 = 3 + 4;
    //sayi1 in değeri 7

    //istersen bir sayı ve bir değişken ile tanımlarız
    int sayi2 = sayi1 + 25;
    //sayi2 nin değeri 32

    //istersek iki değişkenin toplamı olarak yazabilirz
    int sayi3 = sayi1 + sayi2;
    //sayi3 ün değeri 39

    //ikiden fazla operant da kullanabiliriz
    int sayi4 = sayi3 + sayi2 + 33 + sayi1;
    //sayi4 ün değeri 111

    return 0;
}
```

ÇOOOOK ÖNEMLİ

= işareti, matematikteki eşittir işareti değildir. = operatörüne atama operatörü denir.C , = işaretinin sağ tarafındaki işlemi yapar, çıkan sonucu sol taraftaki değişkene atar.Bu sayede:

```c
#include <stdio.h>

int main(){

   int sayi = 100;

   sayi = sayi + 25;

    return 0;
}
```

şeklinde bir atama yapabiliriz. Sayi değişkenine atama yapılan satır:

sayi = 100 + 25 ; olarak yorumlanır ve sayi değişkeninin yeni değeri 125 olur.

Bununla birlikte

int bir_sayi = bir_sayi + 50; şeklinde tanımlama yapamayız çünkü önce sağ taraf çalışır ve henüz bir_sayi isimde bir değişken olmadığı için bu şekilde değişken tanımlayamayız.

Diyelim ki bir değişkenin değerini 1 arttırmak istiyoruz bu işlemi:

```
int sayi = 10;

sayi = sayi + 1;
```

Şeklinde yapabiliriz, ya da

```
int sayi = 10;

sayi += 1;
```

şeklinde de yapabiliriz ama geleneksel olarak bir sayiyi 1 arttırmanın yolu:

```
int sayi = 10;

sayi++;
```

değişkenin sonuna ++ işareti koymaktır. Bu yönteme post-increasement da denir.

Bir sayıyı 1 dışında bir sayı kadar arttırırken ,mesela 5, post-increasement yöntemi kullanamayız istersek

```
int sayi = 10;

sayi = sayi + 5;
```

şeklinde uzun yoldan istersek de

```
int sayi = 10;

sayi += 5;
```

şeklinde kısa yoldan işlem yapabiliriz

bu kısa yolu değişkenin değerini, başka bir değişkenin değeri kadar arttırırken de kullanabiliriz.

```
int sayi1 = 10;
int sayi2 = 5;

sayi1 += sayi2;
```

Matematiksel işlemleri printf içinde de kullanabiliriz

```
printf("%d\n",5 + 3);
```

çıktı: 8

ÇIKARMA

- işareti kullanılır. Kullanımı toplama işlemine benzer

```
//sayılar ile işlem yapma
int sayi1 = 20 - 10;

//bir değişken ve bir sayı
int sayi2 = sayi1 - 5;

//iki tane değişken
int sayi3 = sayi1 - sayi2;
```

Negatif sayı ile tanımla veya atama yaparken kullanılabilir

```
int sayi = -10;

int sayi2 = -sayi;
//sayi2 nin değeri 10
```

```
int sayi = 10;
int sayi2 = 5;

//değişkenin değerini bir değişken kadar azaltma
sayi = sayi - sayi2;

sayi -= sayi2;
```

```
int sayi = 10;

//değişkenin değerini bir sayi kadar azaltma
sayi = sayi - 5;

sayi -= 5;
```

ÇARPMA

* (yıldız – asterisk) işareti kullanılır.

```
int sayi = 10 * 10;

//uzun yoldan bir sayı ile çarpma
sayi = sayi * 5;

//kısa yoldan bir sayı ile çarpma
sayi *= 2;

//kısa yolu değişkenlerle çarpma için de kullanabiliriz
int sayi1 = 25;
int sayi2 = 4;

sayi1 *= sayi2;


/*tıpkı diğer operatörler gibi bir değişkeni başka
 değişkenlerin çarpımı olarak da yazabilirz*/
 int sayi1 = 20;

 int sayi2 = sayi1 * 5;

sayi**;
/* yazımı geçersizdir, bu yöntemi sadece + ve -
için kullanabiliriz*/
```

NOT:

Matematikteki işlem önceliği (operator precedence) programada da geçerlidir. Çarpma ve toplama işleminin yan yana olduğu durumlarda önce çarpma işlemi yapılır.

int sayi = 2 + 5 * 3;

şeklinde yapılan bir tanımlamada sayi değişkeninin sonucu 17 olacaktır ;21 değil. Çünkü bu kod satırır aşağıdaki gibi işler:

2 + 5 * 3
2 + 15
17

Eğer sonucun 21 çıkmasını yani 2 ile 5 i toplayıp 3 ile çarpmak istersek tıpkı matekatikteki gibi parantez kullanırız.

```
int i = (2 + 5)  * 3;
        7 * 3
        21
```

Gelin bu öğrendiklerimizle 2 tane soru çözelim:

Soru1: kullanıcan yarıçapını aldığınız dairenin çevresini ve alanını bulun

```c
#include <stdio.h>

int main(){

    double pi = 3.141;

    /*yarıçap değeri ondalıklı sayı olabilir. Bu nedenle
    int yerine double kullanmak daha uygun*/
    double yaricap;
    printf("yaricap: ");
    scanf("%lf",&yaricap);

    /* tamsayı ile ondalık sayıyı çarparsak sonuç
    ondalıklı sayı çıkar bu yüzden çevre ve alan
    değişkenlerini double olarak tanımlıyoruz*/
    double cevre = 2 * pi * yaricap;
    double alan = pi * yaricap * yaricap;

    printf("Cevre: %f\nAlan: %f", cevre, alan);

    return 0;
}
```

Soru2: Uzun ve kısa kenar uzunluklarını input olarak aldığınız dikdörtgenin çevre ve alanını bulun.

```c
#include <stdio.h>

int main(){

    double kisa, uzun;
    printf("uzun ve kisa kenari sirayla girin: ");
    scanf("%lf %lf",&uzun, &kisa);

    double cevre = 2 * (kisa + uzun);
    // cevre = 2 * uzun + 2 * kisa; olarak da yazılabilirdi

    double alan = uzun * kisa;

    printf("Cevre: %f\nAlan: %f\n", cevre, alan);

    return 0;
}
```

BÖLME

/ işareti kullanılır fakat çalışma stili birazcık farklıdır

int / int bölmesi yaparsak (ki buna integer division denir) eğer ; sonuç, integer (tamsayı) çıkar (bölme işlemindeki bölümün ondalıklı kısmını alamayız) yani :

10 / 3 işleminin sonucu 3.33333... değil, sadece 3 tür.

Eğer ondalıklı kısmı istiyorsak iki sayıdan en az bir tanesini ondalık sayı yani:

double / int , int / double , double / int şeklinde yapmalıyız.(double yerine float da olur)

10.0 / 3
10 / 3.0 ifadeleri bize 3.33333 ile başlayan sonucu verecektir
10.0 / 3.0

bu durumu iki klasik soru ile inceleyelim:
SORU 2: kullanıcıdan alınan üç tamsayının ortalamasını bulunuz.

```c
#include <stdio.h>

int main(){

    int s1, s2,s3;
    printf("sayiları giriniz: ");
    scanf("%d %d %d",&s1, &s2, &s3);

    //ortalamanın ondalıklı sayı çıkma ihtimali var
    double ortalama = (s1 + s2 + s3) / 3;
    //bu satırda bir hata var gibi

    printf("%f\n",ortalama);

    return 0;
}
```

Verdiğimiz sayılar 5 10 ve 20 olsun. Ortalaması 11.66666666 ama çıktı olarak 11.000000 alıyoruz. ÇÜNKÜ: s1 s2 ve s3 birer integer. Üç integerin toplamı da integer oluyor, bölen sayı da int olduğu için C sonuç değerini 11,666666 değil 11 olarak döndürüyor. ortalama değişkeni de ondalıklı sayı tutmak istediği için 11 sayısını 11.000000 a çeviriyor.

Bu durumdan kurtulmak için
ya 11. satırı sağdaki gibi
tanımlayacağız ya da

```c
double ortalama = (s1 + s2 + s3) / 3.0;
```

Ondalıklı sayı + tamsayı = ondalıklı sayı olduğu için s1 s2 veya s3 değişkenlerinden en az birini double veya float olarak tanımlayacağız

SORU 2:Aşağıdaki formulü kullanarak kullanıcıdan alınan fahrenheit değerinin selsiyusa çeviriniz.

C = (5 / 9) * F

```c
#include <stdio.h>

int main(){

    int sicaklik;
    printf("fahrenheit: ");
    scanf("%d",&sicaklik);

    //formüldeki 5/9 selsiyusu ondalık sayı yapabilir
    //bu yüzden selsiyusu double olarak tanımlıyoruz
    double selsiyus = (5 / 9) * sicaklik;

    printf("%f\n",selsiyus);

    return 0;
}
```

İnput olarak ne verirsek verelim sonuç hep 0.000000 çıkacak 11. satıra bakalım:
5 / 9 kesrinin değeri 0.555555556 dır fakat 5 ve 9 integer olduğu için c, sonucu 0 olarak alır 0 ile ne çarparsak çarpalım sonuç 0 çıkar.

Bundan kurtulmak için (5 / 9) yerine (5.0 / 9) yazmamız yeterli olacaktır

Aklınızda “selsiyusu double olarak tanımlamamıza rağmen neden int / int bölmesi oluyor” sorusu varsa = işaretinin sağdan sola çalıştığını hatırlayın. double keyword’ü eşitliğin sağ tarafını kapsamaz. Sağ tarafta ise, parantez olduğu için ilk önce (5 / 9) işlemi yapılacak ve sonuç 0 bulunacaktır. double keyword’ünün yapabileceği tek şey 0’ ı 0.000000 yapmaktır.

KALAN BULMA

% (modulus) operatörü ile kalan bulma işlemi yapılır

```
int sayi = 10 % 6;
//sayinin değeri 4

int sayi2 = 35 % sayi;
//sayi2 nin değeri 3

//kısa yol olarak
int bir_sayi = 100;
bir_sayi %= 5;
//bir_sayi değişkeninin değeri 0
```

% operatörünün en önemli özelliği ise operantlarının ikisi de integer olmak zorundadır. Aşağıdaki koda bakalım:

```
C a.c > ...
1   #include <stdio.h>
2
3   int main(){
4
5       int sayi = 10.0 % 6;
6
7       return 0;
8   }
```

10.0 bir integer değil

Kodumuzu çalıştırmayı deneyelim:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:5:21: error: invalid operands to binary % (have ‘double’ and ‘int’)
     int sayi = 10.0 % 6;
                     ^
```

hata: % operatörüne geçersiz operantlar (double ve int almış)
gördüğünüz gibi % operatörünü sadece integer ile kullanılıyor

Operantların ikisi de int tipinde olmak koşuluyla değişken olabilir:

```
int sayi = 25;
int sayi2 = 8;

int sayi3 = sayi % sayi2;
//sayi3 ün değeri 1 dir
```

Şimdi % operatörünün 2 yaygın kullanımını görelim:

SORU 1:Bir sayının tek veya çift olduğuna nasıl karar verirsiniz?

Eğer bir sayının 2 ile bölümünden kalan 0 ise o sayı çifttir. Kalan 1 ise sayı tektir.

```c
int sayi = 25;

printf("%d",sayi % 2);
/*virgül de bir operatördür sağdan sola doğru
çalışır. 25 in 2 ile bölümünden kalan 1 dir*/
```

Çıktı: 1
25 bir tek sayıdır

SORU 2:Bir sayının onlar birler ve yüzler basamağındaki rakamları nasıl bulursunuz:
birler basamağı sayının 10 ile
onlar basamağı sayının 100 ile bölümünden kalandır.
yüzler basamağı sayının 1000 ile

```c
int sayi = 1234;

int birler = sayi % 10;//4
int onlar = sayi % 100;//3
int yüzler = sayi % 100;//2
```

Başka bir yöntem daha var:

```c
int sayi = 1234;

int birler = sayi % 10;//4
sayi /= 10;//artik sayi 123

int onlar = sayi % 10;//123 ün 10 ile bölümünden kalan 3
sayi /= 10;//artik sayi 12

int yüzler = sayi % 10;//12nin 10 ile bölümünden kalan 2
```

gördüğünüz gibi sayının 10 ile bölümünden kalanı alıp 10 a böldükten sonra tekrar 10 bölerek gidersek basamaklardaki rakamlara erişebiliriz

TYPE CASTİNG

Type casting (tip dönüşümü) bir veri tipinin değerini başka bir veri tipine dönüştürmektir.
İki tür tip dönüşümü vardır:
1)İmplicit type casting (kapalı tip dönüşümü)
2)Explicit type casting (açık tip dönüşümü)
Bunların ne olduğunu örnekler üzerinden açıklayalım:

IMPLICIT TYPE CASTING

Herhangi bir ek kelimeye ihtiyaç duymadan dönüşüm yapmaktır.

1)double veya floatı int'e çevirmek

bir ondalık sayı değerini bir tamsayıya atarsak ne olur

```
int sayi = 3.75;

printf("%d\n",sayi);
```

Çıktı: 3

yani int tipine bir ondalık sayı atadığımızda sadece tam sayı kısmını atadık. Burada çok önemli bir şeyi fark etmeliyiz: VERİ KAYBETME İHTİMALİMİZ VAR. ondalık sayının virgüllü kısmını kaybediyoruz. Ondalık kısmın 0 olduğu durumda bir kaybımız yok ama bunun dışındaki durumlarda dikkatli olalıyız

2)int tipini double veya floata çevirmek

bir tam sayı değerini float veya double ' a atarsak ne olur

```
double sayi = 3;

printf("%f\n",sayi);
```

Çıktı: 3.000000

yani 3 sayısı ondalık sayı karşılığı olan 3.0 a çevriliyor.

Peki ya aşağıdaki gibi bir satırda ne olur:

```
double sayi = 3 + 4.5;
```

int + double işlemi görüyoruz. 3. derste veri tiplerinin değer aralıklarından bahsetmiştik. C kendini güvence altına almak ister; eğer aralığı farklı iki tane veri tipi varsa, aralığı küçük olanı aralığı büyük olana çevirir yani bu satır:

```
double sayi = 3.0 + 4.5;
```

Olarak işlem görür. Bu olayın değişkenlerde nasıl işlediğini görelim

```
int tamsayi = 5;
double ondalik_sayi = 3.14;

double sayi = tamsayi + ondalik_sayi;
```

int + double işlemi yapıyoruz yani int tipinden double tipine dönüşüm olacak. Burada bilmemiz gereken şey su:

tamsayi değişkeninin değerini 5.0 yapmıyoruz çünkü biz her ne kadar değişken isimleriyle toplama işlemi yapıyor gibi gözüksek de C değerler üzerinden işlem yapıyor yani sadece "tamsayi" yazan yere 5, "ondalik_sayi" yazan yere de 3.14 yaziyor . 3. ü satırı C:

```
double sayi = 5 + 3.14;
```

şeklinde görüyor. Sonra da 5.0 + 3.14 işlemi yapıyor.

yaptığımız örneklerde int tipi aralığı daha geniş olan double tipine terfi etti. Bu terfi işlemine "data promotion" denir.

Gelin aralıkları sıralayalım:

Yüksek
double
float
long int
int
Alçak

Bu tablodan örnek olarak çıkarabileceklerimiz:

int ile long int işleme girdiğinde int değeri long int tipindeki karşılığına dönüşür

float ile double tiplerinden değişkenler işleme girdiğinde float double tipine çevrilir.

Char ile başka bir derste ilgileneceğiz

EXPLICIT TYPE CASTING

Kapalı tip dönüşümünden farklı olarak ek bir sözcük kullanarak tip dönüşümü yapmamızdır. Bunun için dönüşüm yapmak istediğimiz değişkenin ya da sayının yanına parantez içinde, dönüştürmek istediğimiz veri tipini yazarız

```
double sayi = (double) 3 ;
```

ilk önce sağ taraf çalışır ve 3 , 3.000000 a dönüşür daha sonr abu ondalıklı değer sayi değişkenin içine atanır. Başka bir örnek:

```
double sayi = 2.71;

double sayi2 = (int) sayi;
```

Bu kod parçasında sayi değişkeninin değeri olan 2.71, tamsayıya (2) çevrilir ve sayi2'ye atanır. sayi2 değişkenin değeri 2.000000 olur.

Peki aşağıdaki kodda ne olur

```
double sayi = 2.71;
double sayi2 = 3.14;

double sayi3 = (int) sayi + sayi2;
```

sayi3 ün değeri 5.14 olur çünkü (int) ifadesi sadece yanındaki sayıyı tamsayıya çevirir. Yani 3. satırı C

```
double sayi3 = 2 + 3.14;
```
Olarak görür.

Eğer iki ondalık sayıyı toplayıp integer'a çevirmek istersek parantez kullanırız.

```
double sayi = 2.71;
double sayi2 = 3.14;

double sayi3 = (int) (sayi + sayi2);
```

Paranteze ihtiyaç duyduğumuza göre (type) için de bir işlem önceliğinden bahsedebiliriz. Şu ana kadarki operatörlerin işlem önceliği sıralamasını yapalım.

()

(type)

* / %

+ -

```
double ortalama = (double)(s1 + s2 + s3) / 3;
```

Önceki derslerdeki ortalama bu şekilde de hesapnabilir

ASCII

Bu derste önceden yapmadığımız char dönüşümlerini yapacağız:

Önce bir rakamı char tipine çevireceğiz

```
//rakamı char tipine çevirmek
int sayi = 5;
//bu 5 i '5' yapmak istiyoruz

char tek_tirnak_icinde_sayi = sayi + '0';
```

Harika, bununla birlikte bu yöntem sadece rakamlarda çalışır. Eğer sayi değişkeninin değerini 55 olarak verip tek_tırnak_icinde_ sayi değişkenini bastırırsak

```
#include <stdio.h>

int main(){

    int sayi = 55;

    char tek_tirnak_icinde_sayi = sayi + '0';

    printf("%c\n",tek_tirnak_icinde_sayi);


    return 0;
}
```

Çıktı olarak g harfini alıyoruz. NEDEN?

KARŞINIZDA ASCII TABLOSU

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Bu tablodan çıkaracağımız bir şey var: BÜTÜN CHAR DEĞERLERİNİN İNTEGER OLARAK BİR KARŞILIĞI VARDIR. Bu karşılıkların değer aralığı -128 ile 127 arasındadır fakat BİZ TABLODAN GÖRDÜĞÜMÜZ GİBİ 32 127 ARASI İLE İLGİLENECEĞİZ.

Önceden yaptığımız tip dönüşüm tablosunda en aşağıda int vardı. Char veri tipi onun da aşağısında. İlk yazdığımız kodu tekrar inceleyelim:

```
//rakamı char tipine çevirmek
int sayi = 5;
//bu 5 i '5' yapmak istiyoruz

char tek_tirnak_icinde_sayi = sayi + '0';
```

sayi + ‘0’ ifadesinde char ve int var. tip dönüşümü olacak ve char veri tipinden olan int’e dönüşecek(terfi edecek).

‘0’ char’ının int karşılığı 48 öyleyse

```
char tek_tirnak_icinde_sayi = sayi + '0';
```

satırı
5 + 48; olarak işlem görecek ve

char tek_tirnak_icinde_sayi = 53;

şeklinde yorumlanacak. 53 sayısını da değişkenin veri tipi char olduğu için chara çevireceğiz ve 53 ün char olarak karşılığı ‘5’ .

Diğer kodda ise

```c
#include <stdio.h>

int main(){

    int sayi = 55;

    char tek_tirnak_icinde_sayi = sayi + '0';

    printf("%c\n",tek_tirnak_icinde_sayi);


    return 0;
}
```

ise char tanımladığımız satır 55 + 48 = 103 olarak yorumlanacak ve değişkene 103’ ün char karşılığı olan ‘g’ karakteri atanacak.

Şimdi ise bir char tipinde yazılan bir rakamı int tipine dönüştürelim.
Chardan 48 çıkarmak yeterli olacaktır. Çünkü tabloda sayılar 48den başlıyor.

```c
char tek_tirnak_icinde_sayi = '7';
//'7' yi 7 yapmak istiyoruz

int int_tipine_donusmus_hali = tek_tirnak_icinde_sayi - 48;
```

Aşağıdaki satır 55-48 = 7 olarak yorumlanıyor.

Son olarak: iki basamaklı sayıyı char tipine dönüştüremeyiz çünkü char 1 tane karakter tutabilir. Aynı kuraldan dolayı double tipini de char a çeviremeyiz çünkü bir onalık sayıda da birden fazla rakam var. tırnak içindeki bir rakamı ondalık sayıya çevirmek için
char -> int -> double dönüşümü yapmak gerekir.

Charın int değerini
printf(“%c”,char_degiskeni); ile yazdırabiliriz

const keyword

tanımladığımız bir değişkeni atamaya yani sonradan değiştirmeye kapatmak istiyorsak ya da kodu okuyan kişiye o değişkenin değişmemesi gerektiğini anlatmak istiyorsak const keyword'ünü kullanırız. Bu şekilde "değişmeyen değişken" tanımlarız.

Kullanımı:

const veri_tipi değişken_ismi = başlangıç_değeri;

```
const double pi = 3.14159;
```

Eğer bu değişkene atama yapmak istersek:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function 'main':
a.c:7:7: error: assignment of read-only variable 'pi'
    pi = 2.71;
       ^
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$
```

Hata alırız ve kod çalışmaz.
Hata: sadece okunabilir "pi" değişkenine atama
Hatadan anladığımız kadarıyla bu değişkenle atamayız sadece değerini kullanabiliriz:

```
const double pi = 3.14159;

double cevre = 2 * pi * yaricap;
```
gibi.

Const ile değişmeyen değişken tanımlıyoruz. buna benzeyen bir önişlemci komutu var

#define

```
#include <stdio.h>

#define PI 3.14159

int main(){

   printf("%lf",PI);

    return 0;
}
```

işaretinden anladığımız #define komutunun bir önişlemci komutu olmasıdır.

Define kelimesinin Türkçesi "tanımla" demektir fakat bu sizi yanıltmasın bu satırda tanımlamıyoruz. Önişlemciyi

Define kelimesinin Türkçesi "tanımla" demektir fakat bu sizi yanıltmasın. Bu satırda DEĞİŞKEN TANIMLAMIYORUZ. Önişlemciyi bir text editör (yazı düzenlemeye yarayan bir araç) olarak düşünebilirsiniz. ÖNİŞLEMCİNİN YAPTIĞI ŞEY PI YERİNE 3.14159 YAZMAK

Bu yüzden PI ifadesinin yanına double yazmıyoruz. Bu işlem de “değişmeyen değişken” tanımlamaya benziyor çünkü PI ifadesine yani 3.14159 a (mantıken) bir sayı atayamayız. Atama değişkenlerle yapılır:

```c
#include <stdio.h>

#define PI 3.14159

int main(){

    PI = 2.71;

    return 0;
}
```

Çalıştırırsak:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:7:8: error: lvalue required as left operand of assignment
     PI = 2.71;
        ^
```

hata: atama yapılması için sol tarafa lvalue(sol tarafa yazılıp atama yapılmaya uygun bir ifade) gerekli.

pow - sqrt

pow(double taban,double üs) fonksiyonu üs almaya
sqrt(double sayi) fonksiyonu karekök bulmaya yarar.

Bu fonksiyonları kullanmak için math.h kütüphanesini eklemeliyiz Bunu

#include <math.h> satırını programın başına yazıyoruz.

İki fonksiyonda içine sayı olarak double türünden sayı alır ve sonuç olarak bir double sayı verir.

```c
#include <stdio.h>
#include <math.h>

int main(){

    printf("5 in 3. kuvveti: %f\n",pow(5.0, 3));
    printf("25 in karekökü %f\n",sqrt(25));

    return 0;
}
```

Bu kodu devc++ ta normal bir şekilde çalıştırabiliriz ama linux kullanıcıları ve derleyici komutları derleyen kullanıcılar gcc ifadesinden sonra (-lm) ifadesini eklemelidir. Bu ifade linkera math.h kütüphanesini eklemeyi emreder.

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc -lm a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ ./a
5 in 3. kuvveti: 125.000000
25 in karekökü 5.000000
```

Başta bu fonksiyonların double alacağını söylememize rağmen int değerleri de verdik (3 ve 25). Hiç problem değil bu fonksiyonlar kendi içinde tip dönüşümü yapar ve bu sayıları double’a çevirir.

Ek bilgi: Fonksiyonların içlerine aldığı değerlere tip dönüşümü yapmasına “coercion of arguements” denir.

SORU: kullanıcan iki tane nokta kordinati alın ve iki nokta arası uzaklığı bulun.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

```c
#include <stdio.h>
#include <math.h>

int main(){

    double nokta1_x, nokta1_y;
    double nokta2_x, nokta2_y;

    printf("ilk noktanin kordinatlarini x y seklinde girin");
    scanf("%lf %lf",&nokta1_x, &nokta1_y);

    printf("ilk noktanin kordinatlarini x y seklinde girin");
    scanf("%lf %lf",&nokta2_x, &nokta2_y);

    double uzaklik = sqrt(pow(nokta1_x - nokta2_x, 2) + pow(nokta1_y - nokta2_y, 2));

    printf("uzaklik: %f\n",uzaklik);

    return 0;
}
```

Eğer:
undefined referance to pow
undefined referance to sqrt

hatası alırsanız kodunuzu

gcc a.c -lm -o a

komutuyla derlemeyi deneyin

Son olarak: Eğer ondalıklı sayının sadece virgülden sonraki 2 basamağı ile yazdırmak isterseniz

`printf("uzaklik: %.2f\n",uzaklik);` kodunu kullanabilirsiniz.

Eğer sadece tam kısmı istiyorsanız

`printf("uzaklik: %.f\n",uzaklik);` Kodunu kullanabilirsiniz
Bu gösterimler hakkında ayrıntılı bilgiyi kursun ilerleyen kısımlarında göreceğiz.

SCANF'İM NEDEN ÇALIŞMIYOR:

```c
#include <stdio.h>

int main(){

    int sayi;
    char harf;

    printf("bir sayi girin: ");
    scanf("%d",&sayi);

    printf("bir harf girin: ");
    scanf("%c",&harf);

    printf("tesekkurler\n");

    return 0;
}
```

Yukarıdaki kod masum bir şeklilde bir sayi bir harf ister gibi görünüyor ama çalıştıp içine " 34 " ve " i " girmek istediğimiz zaman

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ ./a
bir sayi gir: 34
bir harf ver: tesekkurler
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$
```

bizim harf girmemizi beklemiyor ve "tesekkurler" yazdırıyor. Sebebi ile vakit kaybetmek istemiyorum. Bu durumdan kurtulmak için yapmamız gereken şey: bizi beklemeyen scanf içine "%c" yerine " %c" yazmak yani %'den önce boşluk bırakmak:

```c
scanf(" %c",&harf);
```

OVERFLOW ERROR – TAŞMA HATASI

int değişkenine tutamayacağı bir değer atarsak ne olur. Mesela int tipine(sınırı yaklaşık olarak 2 milyon) 10 milyon atasak:

```c
#include <stdio.h>

int main(){

    int sayi = 10000000000;

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

    return 0;
}
```

Bu kodu çalıştırmayı deneyelim:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:6:16: warning: overflow in implicit constant conversion [-Woverflow]
     int sayi = 10000000000;
                ^~~~~~~~~~
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ ./a
1410065408
```

bir war ning yani uyarı alıyoruz. Uyarının errordan farklı error veren programı çalıştıramazsınız buna karşılık warning veren kodu çalıştırabilirsiniz AMA sonucunuz çok büyük ihtimalle yanlış çıkar(ben daha uyarıya rağmen düzgün çalışan bir kod yazmadım)

Uyarıya rağmen çalıştırdık ve hiç işe yaramayacak bir sayı aldık.

Ek bilgi sayiya atanan değer = 1000000000 % (int tipinin max değeri + 1) bunu bilmeseniz de olur. Ama işe yaramayan bir sonuç alırsanız bu ders aklınızda olsun. Bu olay atama yaparken de gerçekleşebilir:

sayi = 9999999* 878887 * 665 tarzı bir ifadede de sayiya sağdaki işlemin modu atanır ve işe yaramayan bir sayı elde ederiz.

ŞİMDİ GÖRECEĞİNİZ HATA MESAJLARI LİNUX SİSTEMİNDE ALINMIŞ HATALARDIR. WİNDOWSTA ALACAĞIMIZ HATALARI VİDEO DERSTE GÖRECEĞİZ.

Şimdi en sık yapılabilecek hataları inceleyeceğiz. Aşağıdaki kodu inceleyin. Anlamasanız da olur kodun ne yaptığıyla ilgilenmeyeceğiz. Ve bu hataları ezberlemeye çalışmayın hataları anlatmamın sebebi bir hatanın sebebini bulamadığınız zamanlar için size kaynak verip zaman kazandırmak

```c
#include <stdio.h>

int main(){

    int i = 5;
    int j = 15;

    int k = i * j;

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

    return 0;
}
```

1. satırda include işlemini yapmazsak

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:9:5: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]
     printf("%d\n",k);
     ^~~~~~
a.c:9:5: warning: incompatible implicit declaration of built-in function ‘printf’
a.c:9:5: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’
```

uyarı diyor ki printf tanımlı değil. Note kısmı da stdio.h ı include etmemizi söylüyor

---

printf i print şeklinde yanlış yazarsak

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:10:5: warning: implicit declaration of function ‘print’; did you mean ‘printf’? [-Wimplicit-function-declaration]
     print("%d\n",k);
     ^~~~~
     printf
/tmp/cct5PnAX.o: `main' fonksiyonunda:
a.c:(.text+0x32): `print'ye tanımsız başvuru
collect2: error: ld returned 1 exit status
```

diyor ki acaba print i yanlış mı yazdın? Printf demek istemiş olabilir misin aşağıdan 2. satırdaki yazıyı siz “undefined referance to print” olarak görebilirsiniz. Bu uyarı da size yanlış yazdığınızı gösteriyor.

Printfin solundaki parantezi unutursak

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:10:11: error: expected ‘;’ before string constant
     printf"%d\n",k);
           ^~~~~~
a.c:10:19: error: expected statement before ‘)’ token
     printf"%d\n",k);
                  ^
```

Printfin sağındaki parantezi unutursak

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:10:20: error: expected ‘)’ before ‘;’ token
     printf("%d\n",k;
                    ^

a.c:14:1: error: expected ‘;’ before ‘}’ token
 }
 ^
```

---

printf(“%f”,sayi); şeklinde %d yerine %f kullanırsak

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:10:14: warning: format ‘%f’ expects argument of type ‘double’, but argument 2 has type ‘int’ [-Wformat=]
     printf("%f\n",k);
             ~^
             %d
```

en aşağıda zaten %d diye yazıyor

EN YAYGIN HATA “ ; “ UNUTMAK

diyelim ki 5. satırda ; koymayı unuttuk

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:6:5: error: expected ‘,’ or ‘;’ before ‘int’
     int j = 15;
     ^~~
a.c:8:17: error: ‘j’ undeclared (first use in this function)
     int k = i * j;
                 ^

a.c:8:17: note: each undeclared identifier is reported only once for each function it appears in
```

İlk hatada dedi ki “int’ten önce ; gerekli” ve j nin olduğu satırda hata verdi oysa biz i’nin olduğu satırda hata yapmıştık demek ki hatanın gösterildeği satır bizim hata yaptığımız satır olmayabilir

ikinci hatada diyor ki “j tanımlı değil (ilk önce bu değişkeni kullanın)” evet parantez içindeki ifade biraz garip yine de biz buradan tane şey öğreniyoruz:

1)i satırında yaptığımız hata j nin tanımlanmasını engellemiş

2)k değişkeni de j ye bağlı olduğu için tanımlanamıyor

3)bir satırda yaptığımız hata birden fazla error almamıza sebep olabilir.(bir hatayı düzelterek 10+ satır error-warning ten kurtulmuştum)

4)tanımladığımız değişkeni kullanmak zorundayız(bunu çıkarmak birazcık zor ama bu bilgiyi vermenin zamanı geldi diye düşünüyorum)

Şimdi başka bir soruya geçelim: matematiksel olaylarda parantez unutursak nolur

```
int i = (((3 + 9) + 6) * 3;
//en soldaki parantez açık kaldı
//; ü altındaki kırmızılık bunu
//haber ediyor
```

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:5:29: error: expected ‘)’ before ‘;’ token
   int i = (((3 + 9) + 6) * 3;
                             ^

a.c:13:1: error: expected ‘,’ or ‘;’ before ‘}’ token
 }
 ^
a.c:13:1: error: expected declaration or statement at end of input
```

peki scanfteki & ampersantı unutursak ne olur

```
int i ;
scanf("%d",i);
```

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum1$ gcc a.c -o a
a.c: In function ‘main’:
a.c:6:11: warning: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’ [-Wformat=]
   scanf("%d",i);
          ~^
```

Hatırlarsak & koyarak bir değişkenin bellekte ayrılan yerini scanf e veriyorduk bu kodda yeri değil değişkenin kendisini verdik bu yüzden uyarı aldık. “int *” ifadesini pointer bölümünde göreceğiz şimdilik buna takılmayalım.

Son olarak en aşağıdaki return 0; yazarsak başarılı
return 1; yazarsak başarısız çıkış yapmış oluruz. Kod yine aynı şekilde çalışır ama bazen problem bizden hata mesajı yazıp başarız çıkış yapmamızı ister bu durumda return 1; ile çıkış yapmamız gerekir.

ÇALIŞMA SORULARI

SORU1: kullanıcıdan aldığınız sayının küpkökünün 20 fazlasının 5 katını bastırın

kupkok.c

```c
#include <stdio.h>
#include <math.h>

//Kullanıcıdan alınan sayının küpkökünün  20 fazlasının 5 katını bulun.
int main(void){

    int sayi;
    printf("lutfen bir sayi girin");
    scanf("%d", &sayi);

    double sonuc = (pow(sayi, 1/3.0) + 20) * 5;
    //1/3 yazarsak sayi üzeri 0 hep 1 olur

    printf("%.1f\n",sonuc);

    return 0;
}
```

SORU 2 : kilo / boy*boy formülüyle verilerini aldığının kişinin beden kitle indeksini hesaplayın

bki.c

```c
#include <stdio.h>

int main(){

    double boy;
    printf("lutfen boyunuzu metre cinsinden giriniz: ");
    scanf("%lf",&boy);

    int kilo;
    printf("lutfen kilonuzu kg cinsinden giriniz: ");
    scanf("%d",&kilo);

    double bki = kilo / (boy * boy);
    //burda int bölmesi yok çünkü boy double olarak tanımlı

    printf("beden kitle indeksiniz: %f\n",bki);

    return 0;
}
```

SORU 3. kullanıcdan aldığınız 5 basamaklı sayının rakamlarını ayırıp bastırın
12345 -> 1 2 3 4 5 gibi

```c
#include <stdio.h>

int main(){

    int sayi;
    printf("5 basamakli bir sayi girin: ");
    scanf("%d",&sayi);

    int on_binler = sayi / 10000;
    printf("%d ",on_binler);
    sayi %= 10000;

    int binler = sayi / 1000;
    printf("%d ",binler);
    sayi %= 1000;

    int yuzler = sayi / 100;
    printf("%d ",yuzler);
    sayi %= 100;

    int onlar = sayi / 10;
    printf("%d ",onlar);
    sayi %= 10;

    int birler = sayi;
    printf("%d\n",birler);

    return 0;
}
```

SORU 4:Kullanıcıdan aldığınız bin ile 999bin arasındaki sayının uygun kısmına virgül koyunuz
12345 -> 12,345 gibi

```c
#include <stdio.h>

int main(){

    int sayi;
    printf("lutfen 1000 ile 999999 asainda bir sayi girin: ");
    scanf("%d",&sayi);

    int sol = sayi / 1000;
    int sag = sayi % 1000;

    printf("%d,%d\n",sol, sag);

    return 0;
}
```