2) KOŞULLU İFADELER

KOŞULLU İFADE NEDİR

koşullu ifadeler matematiksel karşılaştırmalardır Aşağıdaki örneklere bakalım

3 > 5 ifadesi yanlıştır. C, “yanlış” kelimesini “0” sayısıyla ifade eder yani 3 > 5 ifadesinin değeri 0 dır.

8 < 10 ifadesi ise doğrudur. Bizim doğru kelimemiz C dilinde 0 dışında bir sayı ile ifade edilir. Bu sayı çoğunlukla “1” dir.

= işaretinin matematikteki “eşittir “ değil atama operatörü olduğunu söylemiştik.
İki sayının eşit olup olmadığını karşılaştırmak için “ == ” operatörünü kullanırız.
Sayılar birbirine eşitse koşullu ifadenin değeri 1
sayılar birbirine eşit değilse koşullu ifadenin değeri 0 olur
3 == 3 ifadesinin değeri 1
5 == 6 ifadesinin değeri 0

İki sayı birbirine eşit değil mi diye sormak için “ != “ operatörünü kullanırız
Sayılar birbirine eşitse koşullu ifadenin değeri 0
sayılar birbirine eşit değilse koşullu ifadenin değeri 1 olur
5 != 6 ifadesinin değeri 1
9 != 9 ifadesinin değeri 0

≥ yerine “ >= ”
≤ yerine “ <= “ kullanılır
25 <= 25 değeri 1
25 >= 25 değeri 1
5 <= 9 değeri 1
6 <= 3 değeri 0
30 >= 6 değeri 1
35 >= 100 değeri 0

Bir koşullu ifadenin değerini tersine çevirmek için o ifadeyi paranteze alıp başına “ ! “ koyarız
!(7 == 7) değeri 0 dır: Önce parantez içi çalışır ve 7 == 7 ifadesi 1 olarak belirlenir.
!(1) olur. 1 in tersi de 0 dır.
!(6 < 5) değeri 1 dir

Bunların ne işe yaradığını diğer derste göreceğiz

İF STATEMENT

kullanıcıdan bir sayı alın. Eğer bu sayı 5 ten küçükse ekrana “küçüktür”, 5 ten büyükse “büyüktür” yazınız. Bu soruda iki farklı durum var. Bu iki durumun kodunu yazmak için “if”(ing: eğer) ve “else” (ing:yoksa) keywordlerini kullanacağız.

İlk önce if ve else yapılarının nasıl kodlandığını yazalım:

```
if( koşullu ifade ){

    kodlar

}
else{

    if'in olduğu kısım
    çalışmadığı zaman
    çalışacak kodlar

}
```

Bu yapıyı kendi sorumuza uygulayalım:

koşullu ifade: sayi > 5
bu koşul sağlandığında çalışacak kod:
    printf(“buyuktur\n”);

İF KISMINDAKİ KODLAR İÇİNDEKİ KOŞULLU İFADENİN DEĞERİ 0 DAN FARKLI OLDUĞUNDA YANİ İÇERİDEKİ KOŞUL SAĞLANDIĞINDA ÇALIŞIR

if çalışmadığı zaman yani sayı 5 ten küçük veya 5 e eşit olduğunda çalışacak kod:
    printf(“kucuktur\n”);

```
if( sayi > 5 ){
    printf("buyuktur\n");
}
else{
    prntf("kucuktur\n");
}
```

Süslü parantez kullanmak her zaman tavsiye edilir bununla birlikte iki süslü parantez arasına sadece 1 satır kod yazılacaksa süslü parantez koymak zorunlu değildir. Aşağıdaki kod yukarıdakinin aynısıdır

```
if( sayi > 5 )
    printf("buyuktur\n");

else
    prntf("kucuktur\n");
```

else kısmı zorunlu değildir eğer soru sadece “alınan sayi 5 ten büyükse büyüktür yazdır” olsaydı else kısmını yazmamıza gerek kalmazdı Aşağıdaki kod yeterli olurdu.

```c
#include <stdio.h>

int main(){

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

    if(sayi > 5)
        printf("buyuktur\n");

    return 0;
}
```

Üç tane kolay soru çözelim:
soru1: kullanıcıdan alınan sayı çift ise “çift”,tek ise “tek” yazdırın

a.c > ...

```c
#include <stdio.h>

int main(){

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

    if(sayi % 2 == 0){
        printf("cift\n");
    }
    else{//bir sayının 2 ile böl. kalan ya 0 ya 1 dir
        printf("tek\n");
    }

    return 0;
}
```

soru2: kullanıcıdan alınan 2 sayıdan büyük olanı yazdırın. Sayıların eşit olmadığını varsayın

C karsilastirma.c > ...

```c
#include <stdio.h>

int main(){

    int sayi1, sayi2;
    printf("iki sayi girin: ");
    scanf("%d %d",&sayi1, &sayi2);

    if(sayi1 > sayi2)
        printf("%d\n",sayi1);
    else
        printf("%d\n",sayi2);

    return 0;
}
```

soru3: bir inşaat şirketi 17 rakamını uğursuz görüyor ve asansöre 17 rakamını koymuyor . Asansördeki basılan sayının gerçekte hangi katı gösterdiğini bulun.

C karsilastirma.c > ...

```c
#include <stdio.h>

int main(){

    int buton;
    printf("asansörde hangi sayiya bastiniz: ");
    scanf("%d",&buton);

    int gercek_kat;

    if(buton > 17){
        gercek_kat = buton - 1;
        //18 e bastığında aslında 17 ye gidiyor
    }
    else{
        gercek_kat = buton;
    }

    printf("%d. kata gidiyorsunuz\n",gercek_kat);

    return 0;
}
```

ELSE İF

geçen derste çözdüğümüz sorulardan birini hatırlayalım

kullanıcıdan alınan 2 sayıdan büyük olanı yazdırın. Sayıların eşit olmadığını varsayın

“Sayıların eşit olmadığını varsayın” bunu özellikle söyledik çünkü:
ilk sayının büyük olması 1. durum bu durumu if ile yazabiliriz
ikinci sayının büyük olması 2. durum bu durumu else ile yazdık

sayıların eşit olması ise 3. bir durum. Bu durumu nasıl “else if” ifadesi ile kodlayacağız.

Else if ifadesinin nasıl yazıldığına bakalım

```
if( koşullu ifade ){
    kodlar
}
else if( koşullu ifade ){
    kodlar
}
else{//opsiyonel
    kodlar
}
```

BU KOD NASIL ÇALIŞIR:
Yukarıdan aşağı doğru gidilir

eğer if içindeki koşullu ifade sağlanırsa if bloğu yani if kısmına bağlı olan süslü ifadenin içindeki kodlar çalışır ve else if ve else blokları çalışmaz.

eğer if bloğu çalışmazsa ;else if içindeki koşullu ifadeye bakılır, else if içindeki koşul sağlanıyorsa else if bloğu çalıştırılır. Else bloğu çalıştırılmaz

eğer hem if hem de else if ifadelerindeki koşullu ifadeler sağlanmazsa else çalışır.

NOT:
if else kullanmak için MUTLAKA bir if bloğu gerekir. İf else bloğu kendi başına yazılamaz, bir if bloğu altına konulması gerekir.

NOT:
bir if altına birden fazla else if bloğu konulabilir. Örneklerini çözeceğiz.

şimdi sayıların eşit olabileceği durumu da dahil ederek karşılaştırma sorusunu tekrar çözelim

C elseif.c > ...

```c
#include <stdio.h>

int main(){

    int sayi1, sayi2;
    printf("iki sayi ver");
    scanf("%d %d",&sayi1, &sayi2);

    if(sayi1 == sayi2)
        printf("esitler\n");
    else if(sayi1 > sayi2)
        printf("%d daha buyuk\n",sayi1);
    else
        printf("%d daha büyük\n",sayi2);

    return 0;
}
```

NESTED İF STATEMENTS

bir if bloğunun içine tekrar bir if bloğu koyabiliriz. Aşağıdaki soruyu inceleyelim.

Soru: bir sayi 10'a bölünüyorsa o sayıya "A" sayısı denir. Bir A sayısı aynı zamanda :
21'e bölünüyorsa "AB" sayısı denir
21'e bölünmeyip 7'ye bölünüyorsa o sayıya "AC" sayısı denir
21'e bölünmeyip 3'e bölünüyorsa "AD" sayısı denir.

Buna göre kullanıcıdan alınan sayının A sayıları ailesinden biri olup olmadığını bulun

bir sayının örnek olarak "AC" sayısı olması için ilk önce "A" sayısı olması gerekir. Bu yüzden ilk önce sayının sayısı olup olmadığına bakacağız. Değilse A sayıları ailesinden olmadığını kullanıcıya söyleyeceğiz.

```c
#include <stdio.h>

int main(){

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

    if(sayi % 10 == 0){

        if(sayi % 21 == 0){
            printf("AB\n");
        }
        else if(sayi % 7 == 0){
            printf("AC\n");
        }
        else if(sayi % 3 == 0){
            printf("AD\n");
        }
        else{
            /*21 3 veya 7 ye bölünmüyorsa
            sadece "A" sayısıdır*/
            printf("A\n");
        }
    }
    else{
        printf("bu sayi A sayilari ailesinden degil\n");
    }

    return 0;
}
```

BOOLEAN VARİABLES

Bir koşul sağlandığında değerinin 1, sağlanmadığında 0 olduğunu önceden söylemiştik. Bu 0 ve 1 sayılarının özel bir isimleri vardır:
0 ın özel ismi “false”(tr: yanlış)
1 in özel ismi “true”(tr: doğru)
bu mantıkla; bir if statement’ın içindeki değer “true” olduğunda çalışacağını, “false” olduğunda çalışmayacağını söyleyebiliriz.

False ve true değerleri bazı programlama dillerinde tıpki int veya char gibi ayrı bir veri tipidir fakat C dilinde yukarıda yazdığı gibi sayılarla ifade edilir. Kodda “true” ve “false” ifadelerini kullanabilmek için stdbool.h kütüphanesini eklememiz gerekir. Aşağıdaki kodu inceleyelim

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

int main(){

    if(true){
        printf("hello world\n");
    }

    if(false){
        printf("nasilsin\n");
    }

    return 0;
}
```

yukarıdaki if çalışacak ve hello world yazdırıcak

aşağıdaki if ise çalışmayacak

bu kütüphaneyi dahil etmeden bu iki kelimeyi kullanmak istersek #define kullanacağız

```c
#include <stdio.h>
#define true 1
#define false 0

int main(){

    if(true){
        printf("hello world\n");
    }

    if(false){
        printf("nasilsin\n");
    }

    return 0;
}
```

yukarıdaki kod gibi:
hello world yazılacak
alttaki if çalışmayacak

madem bu ifadeler de aslında birer sayı o zaman parantez içlerine de 1 veya 0 yazabiliriz. fakat bu okunabilirliği azaltır.

bool.c > ...

```c
#include <stdio.h>

int main(){

    if(1){
        printf("hello world\n");
    }

    if(0){
        printf("nasilsin\n");
    }

    return 0;
}
```

En önemli kısım :Değişkenlerin sayı değeri olduğuna göre parantez içine sadece değişkenin kendisini yazabiliriz. O zaman

```c
#include <stdio.h>

int main(){

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

    if(sayi != 0){//sıfıra eşit değilse if bloğu çalışacak
        printf("hello world\n");
    }

    return 0;
}
```

yerine aşağıdaki kodu kullanabiliriz

```c
if(sayi){//sıfıra eşit değilse if bloğu çalışacak
    printf("hello world\n");
}
```

MANTIKSAL OPERATÖRLER

SORU: Kullanıcıdan alınan bir sayının 10 ile 50 arasında olup olmadığını kontrol ediniz

zekice ve masumane şekilde if statement'ı

```c
#include <stdio.h>

int main(){

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

    if( 50 > sayi > 10 )
        printf("evet 10 ile 50 arasında\n");

    else
        printf("degil\n");

    return 0;
}
```

şeklinde yazabiliriz hadi kodu çalıştıralım ve 25 sayısını girelim:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ ./a
lutfen bir sayi giriniz25
degil
```

değil diyor ama 25 10 ile 50 arasında demek ki istediğimiz gibi çalışmayan bir şey var .Hemen açıklayalım

ifin içine tekrar bakın

50 > sayi > 10 KARŞILAŞTIRMA OPERATÖRLERİ SOLDAN SAĞA ÇALIŞIR yani bu satır ilk olarak
50 > sayi
50 > 25 şeklinde çalışır ve bu karşılaştırma doğru olduğu için 50 > sayi ifadesinin 1 dir sağ kısma bakınca elimizde
1 > 10 kalıyor ve bu ifade yanlış olduğu için parantez içindeki koşullu ifadenin değeri 0 "yanlış" oluyor ve if yerine else kısmı çalışıyor. Bu yüzden bu şekilde bir koşullu ifade yazmamalıyız

Bu problemin üstesinden gelebiliriz: bir sayı 10 ile 50 arasında ise 50 den küçük VE 10 dan büyüktür. C "ve" kelimesini "&&" simgesiyle kullanmamıza izin verir. && ile birleştirilen iki koşullu ifadenin olduğunu if bloğunun çalışması için iki ifadenin de "true" olması gerekir: Şimdi kodu tekrar yazalım:

a.c > ...

```c
#include <stdio.h>

int main(){

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

    if( 50 > sayi  && sayi > 10 )
        printf("evet 10 ile 50 arasında\n");

    else
        printf("degil\n");

    return 0;
}
```

C “veya” kelimesini de “||” işaretiyle kullanmamıza izin verir. Veya ile birleştirilen iki koşullu ifadeye sahip if bloğunun çalışması için iki koşuldan 1 tanesinin “true” olması yeterlidir. İkisi doğru ise de çalışacaktır. Gelin bir soru çözelim

soru: kullanıcıdan alınan bir sayının 6 veya 9 a bölünüp bölünmediğini bulun

a.c > ...

```c
#include <stdio.h>

int main(){

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

    if(sayi % 6 == 0 || sayi % 9 == 0 )
        printf("6 veya 9 a bolunuyor\n");

    else
        printf("bolunmuyor\n");

    return 0;
}
```

Kafa karışıklığını önlemek veya işlem önceliğini belirtmek için koşulla ifadeleri parantez içine alabiliriz

soru: 7 veya 5 e bölünen bir sayı aynı zamanda bir çift sayı ise bu sayıya “7C5” sayısı denir. Kullanıcıdan alınan sayının 7C5 sayısı olup olmadığını bulun

a.c > ...

```c
#include <stdio.h>

int main(){

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

    if( (sayi % 5 == 0 || sayi % 7 == 0) && sayi % 2 == 0 )
        printf("7C5 sayisidir\n");

    else
        printf("degildir\n");

    return 0;
}
```

TERNARY (ÜÇLÜ) OPERATOR

ternary operatör karşılaştırma yapan ve diğer operatörlerden farklı olarak 3 tane operant alabilen operatördür “?” ve “:” işaretleri ile kullanılır Yapısını inceleyelim

koşullu ifade ? İfade true ise sonuç : ifade false ise sonuç.

5 sayısı ile karşılaştırma yapan kodu inceleyelim:

```
if( sayi > 5 )
    printf("buyuktur\n");

else
    prntf("kucuktur\n");
```

ternary operator ile bu 4 sarı 1 satıra indirebiliriz.

```
sayi > 5 ? printf("buyuktur\n") : printf("kucuktur\n");
```

Fark ettiyseniz ilk printf’ten sonra ; virgül koymuyoruz.

Farklı kullanımları görmek için 2 örnek yapalım

1)iki sayıdan hangisinin büyük olduğunu ternary operator ile yazalım:

```
printf("%d\n",sayi1 > sayi2 ? sayi1 : sayi2);
```

2)kullanıcıdan alınan sayının mutlak değerini ternary op ile yazdırın

a.c > ...

```
1   #include <stdio.h>
2
3   int main(){
4
5       int sayi;
6       printf("lutfen bir sayi giriniz");
7       scanf("%d",&sayi);
8
9       int mutlak_deger = sayi >= 0 ? sayi : -sayi ;
10
11      printf("%d\n",mutlak_deger);
12
13      return 0;
14
15  }
```

SWİTCH-CASE

switch-case yapısı tıpkı if yapısı gibi farklı durumlarda farklı kodları çalıştırmamızı sağlayan bir yapıdır. Yapısına bakalım

```
switch ( değişken )
{
    case int veya char:

      değişkenin değeri yazılan int veya char olduğunda
      çalışacak kodlar

      break;

    case int veya char:

      değişkenin değeri yazılan int veya char olduğunda
      çalışacak kodlar

      break;

    .
    .istediğimiz kadar case yazabiliriz
    .
    default:

      hiçbir case çalışmayanca çalışacak kodlar
}
```

Bu yapının nasıl çalıştığını ayrıntılarıyla gerçek bir kod üzerinde inceleyelim

a.c > ...

```c
#include <stdio.h>

int main(){

    int sayi;
    printf("1 ile 5 arasında (1,5 dahil) bir sayi girin");
    scanf("%d",&sayi);

    switch( sayi ){

        case 1:
            printf("girdiginiz sayi 1\n");
            break;

        case 2:
            printf("girdiginiz sayi 2\n");
            break;

        case 3:
            printf("girdiginiz sayi 3\n");
            break;

        case 4:
            printf("girdiginiz sayi 4\n");
            break;

        case 5:
            printf("girdiginiz sayi 5\n");
            break;

        default:
            printf("girdiginiz sayi 1 ile 5 arasinda degil");
    }

    return 0;
}
```

Eğer kullanıcı 1 girerse case 1 deki kod çalışır. Break ifadesi ise 9-33. satırlar arasındaki switch bloğundan çıkmamızı sağlar yani break ifadesini görünce diğer case ler çalışmaz

eğer kullanıcı 3 girerse girdiği sayı case 3 e gelinceye kadar önceki caselerin değerleri karşılaştırılır yani 3 ile 1 karşılaştırılır. 3İle 1 eşit olmadığı için bu case geçilir. 3 ile 2 karşılaştırılır . 3 ile 2 farklı olduğu için bu case de geçilir. case 3 e gelince 3 ve 3 birbirine eşit olduğu için case 3 e girilir. Case 3 teki kodlar çalıştırılır ve break ifadesi görülünce switch bloğundan çıkılır.

Eğer kullanıcı 10 girerse 1 2 3 4 5 ile karşılaştırılır ve hiçbirine giremeyeceği için default bölümüne girilir. Default içindeki printf çalıştırılır. Default kısmı son bölüm olup o kısımdan sonra zaten switch bloğundan çıkılacağı için default kısmına break yazmaya gerek yoktur.

break ifadelerini kaldırırsak ne olur.

```
switch( sayi ){

    case 1:
        printf("girdiginiz sayi 1\n");

    case 2:
        printf("girdiginiz sayi 2\n");

    case 3:
        printf("girdiginiz sayi 3\n");

    case 4:
        printf("girdiginiz sayi 4\n");

    case 5:
        printf("girdiginiz sayi 5\n");

    default:
        printf("girdiginiz sayi 1 ile 5 arasinda degil");
}
```

Kodu bu şekilde çalıştırdıktan sonra 3 sayısını verelim:

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ ./a
1 ile 5 arasında (1,5 dahil) bir sayi girin3
girdiginiz sayi 3
girdiginiz sayi 4
girdiginiz sayi 5
girdiginiz sayi 1 ile 5 arasinda degil
```

Neler olduğunu açıklayalım: 3 girdik case 1 ve case 2 yi atladık ve case 3 e girdik ve printf fonksiyonunu çalıştırdık. Case 3 bitmiş olsa da break ifadesi olmadığı için hale switch bloğu içerisindeyiz. Sırada case 4 var ve 4 3'e eşit değil AMA bu gibi break olmadığı durumlarda bir case e girdikten sonra diğer case lerin değerine bakmadan o case lerin içine gireriz. Her hangi bir case sonunda break ifadesiyle karşılaşıncaya kadar ilerlemeye devam ederiz. Ve bu kodda break olmadığı için tüm case lere ve default kısmına girip switch bloğunu terkediyoruz.

Bu durumu avantaja çevirebiliriz
soru: kullanıcıdan yılın kaçıncı ayında doğduğunu alın ve o ayın kaç gün olduğunu yazdırın

```c
int ay;
printf("kacinci ayda dogdunuz: ");
scanf("%d", &ay);

switch(ay){

    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        printf("dogdugunuz ay 31 gun\n");
        break;

    case 2:
        printf("dogdugunuz ay 28 gun\n");
        break;

    case 4:
    case 6:
    case 9:
    case 11:
        printf("dogdugunuz ay 30 gun\n");
        break;
}
```

== YERİNE = YAZMAK

insanlık hali == yerine = yazabiliriz bu durumda ne olacağını inceleyelim.

a.c > main()

```c
#include <stdio.h>

int main(){

    int sayi = 2;

    if( sayi % 2 = 0){
        printf("hello\n");
    }


    return 0;
}
```

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
a.c: In function ‘main’:
a.c:7:18: error: lvalue required as left operand of assignment
     if( sayi % 2 = 0){
                  ^
```

Bu hatayı biliyoruz ama dikkat çekmek istediğim şey şu: == yerine = yazarsak parantez içinde atama yapabiliyoruz.. Sıradaki kodu inceleyelim

```c
int sayi = 2;

if( sayi = 5 ){
    printf("hello\n");
}
```

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ ./a
hello
```

Bu durumda “==” olmadığı c karşılaştırma yapmıyor.
if(sayi = 5) ifadesinde ilk önce parantez içi çalışır ve sayi değişkenine 5 değeri atanır ve kod if(sayi) yani if(5) haline gelir. 5 de 0 haricinde bir değer olduğu için true oluyor. Burada bir error veya warning almadık çünkü C bu atamayı bilerek yaptığımızı düşünüyor.

```c
int sayi = 2;

if( sayi = 0 ){
    printf("hello\n");
}
```

Bu durumda ne olacak?
Sayi nin değeri 0 olacağı için if bloğu çalışmayacak

EN SIK YAPILAN HATA

En sık yapılan hata if statement ın parantezinden sonra ; koymaktır Aşağıdaki koda bakalım

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

int main(){

    if(false);{
        printf("HELLO BEAUTIFUL WORLD\n");
    }

    return 0;
}
```

çalıştıralım

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ ./a
HELLO BEAUTIFUL WORLD
```

if false diyoruz ama çıktı alıyoruz bu kodun nasıl yorumlandığını göstermeme izin verin

```c
if(false)
    ;

{
    printf("HELLO BEAUTIFUL WORLD\n");
}
```

int i = 5; demek gibi bir satıra sadece ; koymak da bir kod satırı yazmaktır. O satır hiçbir şey yapmaz “do nothing statement” olarak geçer. Önceki derslerde sadece 1 satır yazarsak süslü parantez koymaya gerek yok demiştik bu yüzden if statement’a sadece bir “do nothing statement” yani hiçbir şey yapmayan bir kod satırı var.
Aşağıda ise { } süslü parantez içinde bir kod satırı var bu kod satırı C nin tasarımından dolayı çalışıyor .
Sonuç olarak bir if statementtan sonra bir ; koyarsak içindeki değer ne olursa olsun parantez içindeki kod çalışacaktır.

Diğer hata ise parantezi kapatmayı unutmak:

```c
if(true {
```

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum2$ gcc a.c -o a
a.c: In function ‘main’:
a.c:6:13: error: expected ‘)’ before ‘{’ token
     if(true {
             ^
a.c:12:1: error: expected expression before ‘}’ token
 }
 ^
```