**if Deyimi**

if deyiminin geçen biçimi şöyledir:

if(<int türden ifade>)

<deyim>

[else]

<deyim>

**Anahtar Notlar:** if parantezinin içi C# ve Java gibi dillerde boolean türünden olmalıdır. Ancak C’de böyle bir zorunluluk yoktur.

C dilinde if anahtar sözcüğünden sonraki parantez içerisindeki boolean türden bir ifade bulunmak zorunda değildir. if deyimi doğruysa ve yanlışsa kısmından oluşur. Her iki kısımda da bir deyim bulunmak zorundadır. Bu deyim herhangi bir deyim olabilir. Kısacası bu konuda herhangi bir sınırlama yoktur. if deyiminin kendisi dışarıdan bakıldığında tek bir deyimdir. Örneğin:

ifade1;

if (x > 0) {

ifade2;

ifade3;

}

else {

ifade4;

ifade5;

}

ifade6;

Burada toplam 3 deyim vardır.

if deyimi çalışma sistemi şöyledir: Önce if parantezi içerisindeki ifadenin değeri hesaplanır. Bu değer true ise doğruysa kısmındaki(true bölümündeki) deyim, false ise yanlışsa kısmındaki(False bölümündeki) deyim çalıştırılır. Bundan sonra if deyiminin çalışması biter.

if deyiminde else kısmı olmak zorunda değildir. Örneğin:

#include <stdio.h>

int main(void)

{

int val;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&val);

if(val > 0)

printf(“Pozitif”);

printf(“if sonrasi”);

}

Derleyici if deyiminin doğru kısmından sonra else anahtar sözcüğü gelmiş mi diye bakar. Eğer gelmemişse bunun else kısmı olmayan bir if olduğuna karar verir.

Bazen if deyimi yanlışlıkla boş deyim ile kapatılabilir. Bu genelde programcının her ifadenin sonuna noktalı virgül koymaya alışmasından kaynaklanır. Örneğin:

if(ifade);

ifade1;

Bu durumda herhangi bir hata oluşmaz ancak bu bir programlama hatasıdır. Bu durumda kod hem anlamsızlaşır hem de beklenildiği gibi çalışmaz.

Aşağıdaki gibi bir kod syntax olarak doğru olmasına karşın programcı bu biçimde kullanmalıdır:

if(x > 0)

;

else

<deyim>

if deyiminin doğru kısmında boş deyim yazmak “hiçbir şey yapma” anlamına gelir. Bu işlem

if(x <= 0)

<deyim>

şeklinde yapılmalıdır.

#include <stdio.h>

int main(void)

{

int val;

printf("Bir sayi giriniz?\n");

scanf("%d",&val);

if(val > 0)

if (val % 2 == 0)

printf("Pozitif cift sayi\n");

else

printf("Pozitif tek sayi\n");

printf("Programin sonu\n");

}

Bir grup koşuldan biri doğruyken diğerlerinin doğru olma olasılığı yoksa bu koşullara ayrık koşullar denir. Örneğin:

a> 0, a == 0 ve a < 0

koşulları ayrık koşullardır. Ancak:

a> 10 ve a > 6 koşulları ayrık değildir. Ayrık koşulların ayrı if deyimleriyle yapılması her ne kadar syntax olarak yanlış olmasa da kötü bir tekniktir. Örneğin:

#include <stdio.h>

int main(void)

{

int val;

printf("Bir sayi giriniz?\n");

scanf("%d",&val);

if(val > 0)

printf("Pozitif\n");

if(val == 0)

printf("Sifir\n");

if(val < 0)

printf("Negatif");

}

Burada örneğin birinci koşul doğruysa diğerlerine de gereksiz biçimde bakılacaktır. Bu da programın optimizasyonunu ciddi anlamda düşürecektir. Bunun en iyi yöntemi ayrık koşulların else if biçiminde yapılmasıdır. Örneğin:

#include <stdio.h>

int main(void)

{

int val;

printf("Bir sayi giriniz?\n");

scanf("%d",&val);

if(val > 0)

printf("Pozitif\n");

else if(val == 0)

printf("Sifir\n");

else

printf("Negatif");

}

Burada bir else if merdiveni oluşturulmuştur. İç içe if deyimlerinde tek bir else varsa else kısmı içteki if deyimine ait olmaktadır(dangling else). Örneğin:

#include <stdio.h>

int main(void)

{

int val;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&val);

if(val > 0)

if(val % 2 == 0)

printf(“Pozitif cift sayi\n”);

else

printf(“Pozitif tek sayi\n”);

printf(“Programin sonu\n”);

}

Burada bloklama sonrasında artık else kısmı dıştaki if deyimine ait olur. Aşağıdaki gibi bir programda dıştaki if deyiminin else kısmı için herhangi bir bloklama yapmaya gerek yoktur:

#include <stdio.h>

int main(void)

{

int val;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&val);

if(val > 0)

if (val % 2 == 0)

printf(“Pozitif cift sayi\n”);

else

printf(“Pozitif tek sayi\n”);

else

printf(“Pozitif değil\n”);

printf(“Programin sonu\n”);

}

Burada else kısımların hangi if deyimine ait olduğu derleyici tarafından belirlenebilmektedir. Buna göre ilk else kısmı içteki if deyimine ikinci else kısmı ise dıştaki if deyimine aittir.

if deyimi parantezi içerisi C’de int türden olduğu için non-zero yani 0 olmayan tüm değerler için if deyiminin true bölümü çalıştırılır. Örneğin aşağıdaki örnekte ifin true bölümü çalıştırılacaktır:

#include <stdio.h>

int main(void)

{

if(0.5)

printf(“Dogru\n”);

else

printf(“Yanlis\n”);

}

Diğer taraftan 0 false olarak ele alınmaktadır. Bu durum örneğin Javada farklıdır. Java’da if parantezinin içi int türden değil boolean türdendir.

if deyimininde sık yapılan hatalardan biri de karşılaştırma operatörü yerine atama operatörünün kullanılmasıdır. Örneğin:

#include <stdio.h>

int main(void)

{

int val;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&val);

if(a = 10)

{

printf(“Sayi 10\n”);

}

else

{

printf(“Baska bir sayi\n”);

}

}

Bu örnekte val ne girilirse girilsin if deyiminin doğru bölümü çalıştırılacaktır. Bunun nedeni atama operatörünün ürettiği değerdir. Bu yüzden eğer 0’a eşit mi kontrol yapılmak istenseydi ve gene bu hata yapılsaydı bu durumda her zaman if deyiminin yanlış bölümü çalıştırılacaktı. Bu yüzden programcılar bu hatayı önlemek için genelde sabit ifadesinin sola yazarlar aşağıdaki örnekteki gibi:

#include <stdio.h>

int main(void)

{

int val;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&val);

if(10 = a)

{

printf(“Sayi 10\n”);

}

else

{

printf(“Baska bir sayi\n”);

}

}

Çünkü bu durumda kod syntax hatasına neden olucağından programcının hatayı anlaması çok daha kolaydır. Bu kullanım yoda notation (Yoda Notasyonu) olarak isimlendirilir.

**Anahtar Notlar:** if deyimi kullanılarak bir sayının hangi aralıkta olduğu bulunmak isteniyorsa bu aşağıdaki gibi kesinlikle yapılmamalıdır:

#include <stdio.h>

int main(void)

{

int val;

printf("Bir sayi giriniz?\n");

scanf("%d",&val);

if (10 < val < 20) // 1 < 20 Veya 0 < 20

{

printf("Sayi 10 ile 20 arasinda\n");

}

else

{

printf("Sayi 10 ile 20 arasinda degil\n");

}

}

Çünkü bu durumda bu durumda öncelikle 10 < a ifadesinden çıkan 1 veya 0 değeri 20 sabitinden küçük mü kontrolü yapılacaktır. Bu yüzden her zaman if deyiminin doğru bölümü çalıştırılacaktır. Bu kodun doğru yazımı aşağıdaki gibi ve operatörünün kullanılmasıyla yapılmalıdır:

#include <stdio.h>

int main(void)

{

int val;

printf("Bir sayi giriniz?\n");

scanf("%d",&val);

if(10 < val && val < 20)

{

printf("Sayi 10 ile 20 arasinda\n");

}

else

{

printf("Sayi 10 ile 20 arasinda degil\n");

}

}

Örnek Soru: Katsayıları klavyeden girilen bir ikinci dereceden denklemin köklerini bulan fonksiyonu yazınız ve test ediniz.

#include <stdio.h>

#include <math.h>

void findRoots(double a,double b,double c)

{

double delta = pow(b,2) - 4 \* a \* c;

if(delta > 0){

double sqrt\_delta = sqrt(delta);

double x1 = (-b + sqrt\_delta) / (2 \* a);

double x2 = (-b - sqrt\_delta) / (2 \* a);

printf("x1 : %lf\n",x1);

printf("x2 : %lf\n",x2);

}

else if(delta == 0){

double x = -b / (2 \* a);

printf("x1 = x2 = %lf",x);

}

else {

printf("Reel kok yok");

}

}

int main()

{

double a,b,c;

printf("a? b? c?");

scanf("%lf%lf%lf",&a,&b,&c);

findRoots(a,b,c);

}

Başka bir çözüm:

#include <stdio.h>

#include <math.h>

double calculateDelta(double a,double b,double c)

{

return b \* b - 4 \* a \* c;

}

void findRoots(double a, double b, double c)

{

double delta = calculateDelta(a, b, c);

if (delta > 0) {

double sqrt\_delta = sqrt(delta);

double x1 = (-b + sqrt\_delta) / (2 \* a);

double x2 = (-b - sqrt\_delta) / (2 \* a);

printf("x1 : %lf\n",x1);

printf("x2 : %lf\n",x2);

}

else if (delta == 0) {

double x = -b / (2 \* a);

printf("x1 = x2 = %lf",x);

}

else {

printf("Reel kok yok");

}

}

int main()

{

double a,b,c;

printf("a? b? c?");

scanf("%lf%lf%lf",&a,&b,&c);

findRoots(a,b,c);

}