**for Döngüleri**

for döngüleri aslında while döngülerinin daha genel bir biçimidir. for döngü deyiminin genel biçimi şöyledir:

for([ifade1];[ifade2];[ifade3])

<deyim>

for döngüsünün ikinci kısmındaki ifade zero veya non-zero değer üretir kısaca döngünün koşul ifadesi 2.kısmıdır. İki noktalı virgül her zaman parantez içerisinde bulunmak zorundadır. Fakat ifade1, ifade2 ve ifade3 bulunmak zorunda değildir. for döngüsü şöyle çalışır: Döngüye girişte birinci kısımdaki ifade bir kez yapılır. Bir daha yapılmaz. Döngünün yinelenmesinden ikinci kısımdaki ifade sorumludur. Döngü bu ifade non-zero olduğu sürece yinelenir. Üçüncü kısımdaki ifade her döngü deyimi çalıştırıldığında bir kez yapılır. Örneğin:

#include <stdio.h>

int main(void)

{

for(int i = 0;i < 10;++i)

printf(“%d\n”, i);

}

for döngüsünün birinci kısmında değişken bildirim yapılabilir. Fakat diğer kısımlarında yapılamaz. Bildirilen değişkene ilk değer verilmesi zorunludur. Örneğin:

for(int i = 0;i < 10;++i)

<deyim>

for döngüsü en çok aşağıdaki biçimde karşımıza çıkar:

for(ilk değer ; karşılaştırma ; artırım)

<deyim>

Ancak for döngüsünün bölümleri çok daha farklı şekilde de organize edilebilir. Örneğin:  
#include <stdio.h>

int main(void)

{

int i = 0;

for(printf("Birinci kisim\n");i < 10;printf(“ikinci kişim\n”)){

printf(“Deyim”);

++i; }

}

for döngüsünün birinci kısmındaki ifade yazılmayabilir. Bu ifadeyi yukarıya alırsak değişen hiçbir şey olmaz Örneğin:

#include <stdio.h>

int main(void)

{

int i = 0;

for(;i < 10;++i)

{

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

}

}

for döngü deyiminin üçüncü kısmındaki ifade de yazılmayabilir. Bu durumda üçüncü ifade ile yapılması gereken işlem döngü deyiminin sonunda yapılabilir. Örneğin:

#include <stdio.h>

int main(void)

{

for(int i = 0; i < 10;)

{

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

++i;

}

}

Birinci ve üçüncü kısmı boş olan for döngüleri tamamen while döngüleriyle eşdeğerdir:

while(ifade)

<deyim>

ile

for(;ifade;)

<deyim>

tamamen eşdeğerdir. Yani biz while döngüleri for gibi, for döngüleri while gibi kullanılabilir. Örneğin :

for(ifade1;ifade2;ifade3)

<deyim>

ile

ifade1;

while(ifade2)

{

<deyim>

İfade3;

}

döngüleri eşdeğerdir.

for döngüsünün ikinci kısmı da hiç yazılmayabilir. Eğer ikinci kısım yazılmazsa koşulun sürekli sağlandığı kabul edilir. for döngüsünde ikinci kısmın yazılmaması adeta burada 1 değerinin olması gibi bir işlem görmektedir. Yani buraya hiçbir şey yazılmaması ile 1 yazılması tamamen aynı anlamdadır. Örneğin:

#include <stdio.h>

int main(void)

{

for(int i = 0; ;++i)

{

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

}

}

Burada koşul sürekli sağlandığından sonsuz döngü olur.

for döngüsünde üç kısım da boş bırakılabilir. Ancak iki noktalı virgül bulunmak zorundadır:

for(;;){

//..

}

Bazen for döngüleri yanlışlıkla boş deyim ile kapatılabilmektedir:

#include <stdio.h>

int main(void)

{

int i;

for(i = 0;;++i);//Dikkat boş deyim

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

}

Bu durumda herhangi bir hata oluşmaz. Fakat şüphesiz program hatalı çalışır. Bazı durumlarda döngü deyimlerinde programcının algoritması gereği bilerek boş deyim koyması gerekebilir. Bu durumda programcı okunabilirlik açısından noktalı virgülü bir alt satıra koymalıdır:

for(ifade1;ifade2;ifade3)

;

for döngü deyimi ile belirli bir sayıda dönen örnek döngü kalıpları aşağıdaki biçimlerde oluşturulabilir:

1. n bir tamsayı türünden değişken olmak üzere n-kez dönen şu şekilde oluşturulabilir:

for(int i = 0;i < n;++i)

<deyim>

**Örneğin:**

#include <stdio.h>

int main(void)

{

int n = 5;

for(int i = 0;i < n;++i)

printf(“Merhaba\n”);

}

1. n bir tamsayı türünden değişken olmak üzere n-kez dönen döngü şu şekilde oluşturulabilir.

for(int i = n; i >= 0;++i)

<deyim>

Örneğin:

#include <stdio.h>

int main(void)

{

int n = 5;

for(int i = n – 1;i >= 0;--i)

printf(“Merhaba\n”);

}

1. n bir tamsayı türünden değişken olmak üzere n-kez döngü şu şekilde oluşturulabilir:

for(int i = 1; i <= n ; ++i)

<deyim>

Örneğin:

#include <stdio.h>

int main(void)

{

int n = 5;

for(int i = 1;i <= n;++i)

printf(“Merhaba\n”);

}

1. n bir tamsayı türünden değişken olmak üzere n-kez dönen döngü şu şekilde oluşturulabilir:

for(int i = n; i >= 1 ; --i)

<deyim>

Örneğin:  
#include <stdio.h>

int main(void)

{

int n = 5;

for(int i = n; i >= 1;--i)

printf(“Merhaba\n”);

}

**Anahtar Notlar:** Dizilerin ilk elemanlarının indeks numarası 0(sıfır) olduğundan yukarıdaki 1. ve 2. n-kez dönen döngü kalıpları ile bir diziler dolaşılabilmektedir. Diziler(arrays) konusu ileride ele alınacaktır.

for döngüsünün birinci kısmında bildirilen değişkenler döngünün dışında kullanılamazlar. Yalnızca döngünün içerisinde kullanılabilirler. Standartlara göre:

for(bildirim;ifade 2;ifade3)

<deyim>

İşleminin tamamen eşdeğeri şöyledir:

{

Bildirim;

for(; ifade2 ; ifade3)

<deyim>

}

Buna göre ayrı döngülerde aynı isimli değişken bildirimi yapılabilir. Örneğin:

#include <stdio.h>

int main(void)

{

int i;

for(i = 0; i < 10 ; ++i)

printf(“%d\n”, i);

printf(“Dongu sonrasi i : %d\n”,i);

}

Burada i değişkeni döngüden önce bildirilmiştir. Döngüden sonra da kullanılabilir. Şüphesiz yukarıdaki örnekte i değişkenine ilk değer verilip döngünün birinci kısmı boş bırakılabilirdi. Örneğin:

#include <stdio.h>

int main(void)

{

for(int i = 10,k = 1;i > 1;k++,--i)

printf(“(%d,%d)\n”,i,k);

}

**Anahtar Notlar:** C++ ve Java dillerinde de virgül operatörü bulunmaktadır. Ancak Javada C ve C++’ın aksine virgül bir operatör değildir. C/C++ dillerinde virgül son öncelik seviyesinde bulunan ve ürettiği değer her zaman operatörün sağ operandı olan bir operatördür.

**Örnek Soru:** Parametresiyle aldığı int türden bir sayının asal sayı olup olmadığını ger dönen isPrime isimli fonksiyonu yazınız ve test ediniz.

**Çözüm 1(Yavaş Versiyon):**

#include <stdio.h>

#include <stdbool.h>

bool isPrime(int val)

{

if(val <= 1)

return false;

int halfVal = val / 2;

for(int i = 2;i <= halfVal;++i){

if(val % i == 0)

return false;

}

return true;

}

int main(void)

{

for(int i = -10;i < 100;++i){

if(isPrime(i))

printf(“%d ”,i);

}

}

**Çözüm 2(Yavaş Versiyon):**

#include <stdio.h>

#include <stdbool.h>

bool isPrime(int val)

{

if(val <= 1)

return false;

if(val % 2 == 0)

return val == 2;

if(val % 3 == 0)

return val == 3;

if(val % 5 == 0)

return val == 5;

if(val % 7 == 0)

return val == 7;

if(val % 11 == 0)

return val == 11;

for(int i = 13;i \* i < val ;i += 2)

{

if(val % i == 0)

{

return false;

}

}

return true;

}

int main(void)

{

for(int i = -10;i < 100;++i){

if(isPrime(i))

printf(“%d ”,i);

}

}

isPrime örneğinin mülakatların tipik sorularından olduğu unutulmamalıdır. Bu yüzden bu örneğin dikkatli bir şekilde incelenmesi tavsiye edilir.