Döngüler

Belirli işlemlerin tekrarlanmasını sağlayan yapılardır. C dilinde **"for"**, **"while"** ve **"do-while"** döngüsü olarak üç temel döngü türü vardır.

"for" döngüsü, tekrar sayısının bilindiği durumlarda kullanılır. Döngü başında sayaç değişkeni, koşul ve güncelleme işlemleri belirtilir. Genel yazım düzeni şöyledir.

```
for (başlangıç; koşul; güncelleme){
    //
    // işlemler
    //
}
```

"while" ve **"do-while"** döngüleri birbirine çok benzerdirler. **"while"** için önce koşul kontrol edilir sonra döngü bloğunu çalıştırılır. **"do-while"** da ise önce döngü bloğu çalıştırılır sonra koşul kontrol edilir. Koşul sağlanmadığı zaman döngü sona erer. Genel yazım düzenleri şöyledir.

```
while(koşul){
    //
    // işlemler
    //
}
```

```
do{
    //
    // işlemler
    //
} while(koşul);
```

C dilinde verimli ve tekrarlı işlemler yapmak için önemli bir araçtır ve döngüler sıklıkla belirli bir sayıda işlem yapmak, bir dizinin tüm elemanlarını işlemek ve bir koşul gerçekleşene kadar belirli işlemleri yapmak için kullanılırlar.

"for" döngüsü kullanımı

"for" döngüsü, bir işlemi belirli bir sayıda tekrarlamak için kullanılan bir döngü yapısıdır. Döngü, bir başlangıç değeri ile başlar, belirli bir koşul sağlandığı sürece devam eder ve her tekrarda bir güncelleme yapılır. Bu yapı, özellikle belirli bir sayıda işlem yapmak için oldukça kullanışlıdır.

for (başlangıç; koşul; güncelleme) şeklinde yapıda;

- **başlangıç:** Döngünün başında bir kere çalıştırılan kısımdır. Genellikle burada bir sayaç değişkenine başlangıç değeri atanır.

- **koşul:** Döngünün devam etme koşuludur. Bu koşul "true" (doğru) olduğu sürece döngü devam eder. Koşul "false" (yanlış) olduğunda döngü sona erer.

- **güncelleme:** Her döngü sonunda çalıştırılan kısımdır. Genellikle sayaç değişkenini artırma veya azaltma işlemi yapılır.

Örnek 1: 1'den 5'e Kadar Sayıları Yazdıran Döngü

```
// i=1 ile başla, i<=5 olduğu sürece döngü devam etsin,
// her seferinde i'yi 1 artır
for (int i = 1; i <= 5; i++) {
    printf("%d\n", i);     // i'nin güncel değerini ekrana yazdır
 }
```

çıktısı

```
1
2
3
4
5
```

```
// Çift Sayıları Yazdıran Döngü,
for (int i = 2; i <= 10; i += 2) { // her seferinde i'yi 2 artır
    printf("%d\n", i);
 }
```

çıktısı

```
2
4
6
8
10
```

"for" döngüsü kullanımı

Örnek 1: Verilen sayının faktöriyelin hesaplayan döngü

```
int sayi = 5;
int faktoriyel = 1;

for (int i = 0; i <= sayi; i++) {
    faktöriyel *= i;
 }
printf("%d sayının faktöriyeli: %d\n", sayi, faktoriyel);
```

çıktısı

```
5 sayının faktöriyeli: 120
```

Örnek 3: Bir Dizinin Elemanlarını Yazdıran Döngü

Dizilerle **"for"** döngüsünü kullanmak oldukça yaygındır. Aşağıdaki örnekte, "sayilar" adlı bir dizinin tüm elemanları ekrana yazdırılır

```
int sayilar[] = {10, 20, 30, 40, 50};
int uzunluk = sizeof(sayilar) / sizeof(sayilar[0]);
// Dizinin uzunluğunu hesapla  // int uzunluk = 5;

// i=0 ile başla, i dizinin uzunluğundan küçük olduğu sürece devam et
for (int i = 0; i < uzunluk; i++) {
    printf("Element %d: %d\n", i, sayilar[i]);
 }
```

çıktısı

```
Element 0: 10
Element 1: 20
Element 2: 30
Element 3: 40
Element 4: 50
```

"for" döngüsü ve "continue" ve "break" ifadeleri

Döngüyü belirli bir koşul sağlandığında sonlandırmak için **"break"** komutu, döngüdeki belirli bir adımı atlayıp bir sonraki adıma geçmek için ise **"continue"** komutu kullanılır.

```
for (int i = 0; i < 10; i++){
    if (i == 3)
        break;    // i eşit 3 olunca döngüden çık
    printf("%d\n", i);
}
```

çıktısı

```
0
1
2
```

Bu örnekte, i değişkeni 3'e eşit olduğunda **"break"** komutu döngüyü sonlandırır ve artık geri kalan elemanlar işlenmez.

```
for (int i = 0; i < 10; i++){
    if (i == 3)
        continue;  // i eşit 3 ise sonraki adımı atlar
    printf("%d\n", i);
}
```

çıktısı

```
0
1
2
4
5
```

Bu örnekte, i değişkeni 3'e eşit olduğunda **"continue"** komutu döngünün normal adımını atlar ve döngünün bir sonraki elemanı 4 ile devam eder.

İç-içe (nested) "for" döngü yapısı

Bir iç içe "for" döngüsü yapısı, bir "for" döngüsünün içinde başka bir "for" döngüsü kullanılması anlamına gelir. Bu tür yapılar özellikle çok boyutlu veri yapılarıyla çalışırken oldukça yararlıdır. Bir iç içe "for" döngüsünde, dıştaki döngü her çalıştığında, içteki döngünün de tekrar tekrar çalışmasını sağlar.

```
for (dış_değer; dış_koşul; dış_ayar){

    for (iç_değer; iç_koşul; iç_ayar){
        //
        // işlemler
        //
    }
}
```

Aşağıdaki gibi biri sayılardan diğeri harflerden oluşan iki dizi dikkate alalım. Dış dizi sayılardan iç dizi ise harflerden oluşmuş olsun. İki döngü ile bu dizileri ayrı ayrı işleme koyalım.

```
int lst_dis[] = {1, 2, 3};
char lst_ic[]  = {'a', 'b', 'c', '\0'};

for (int i = 0; i < 3; i++) {
    for (int k = 0; lst_ic[k] != '\0'; k++) {

        printf("Dış sayı: %d, iç harf: %c\n", lst_dis[i], lst_ic[k] );
    }
    printf("-----\n");
}
printf("işlem tamamlandı...");
```

çıktısı

```
Dış sayı: 1, iç harf: a
Dış sayı: 1, iç harf: b
Dış sayı: 1, iç harf: c
-----
Dış sayı: 2, iç harf: a
Dış sayı: 2, iç harf: b
Dış sayı: 2, iç harf: c
-----
Dış sayı: 3, iç harf: a
Dış sayı: 3, iç harf: b
Dış sayı: 3, iç harf: c
-----
işlem tamamlandı...
```

İç-içe "for" örnekleri

Çarpım tablosu

Çarpım tablosunu oluşturmak için ilk önce 1'ler, 2'ler, 3'ler gibi dış döngü dizisini oluşturmak gereklidir. Sonra 1'den 10'a kadar iç döngü çarpan dizisi oluşturmalıdır.

```
for (int i = 1; i <= 2; i++) {   // dış döngü

    for (int j = 1; j <= 10; j++) {    // iç döngü

        printf("%2d x %2d = %2d\n", i, j, i * j);
    }
    printf("-----\n");
}
printf("işlem tamamlandı...");
```

çıktısı

```
 1 x  1 =  1
 1 x  2 =  2
 1 x  3 =  3
 1 x  4 =  4
 1 x  5 =  5
 1 x  6 =  6
 1 x  7 =  7
 1 x  8 =  8
 1 x  9 =  9
 1 x 10 = 10
-----
 2 x  1 =  2
 2 x  2 =  4
 2 x  3 =  6
 2 x  4 =  8
 2 x  5 = 10
 2 x  6 = 12
 2 x  7 = 14
 2 x  8 = 16
 2 x  9 = 18
 2 x 10 = 20
-----
işlem tamamlandı...
```

Simge ile dik üçgen oluşturma

Verilen bir tamsayı ile örneğin "*" sembolü ile bir dik üçgen oluşturalım. Verilen sayı dik üçgende hem satır hem de sütun sayısı göstermektedir.

```
int sayi = 5;
for (int i = 1; i <= sayi; i++) {   // satır döngüsü
    for (int j = 1; j <= i; j++) {   // sütun döngüsü

        printf("* ");
    }
    printf("\n");   // bir sonraki satıra geçmek için
}
```

çıktısı

```
*
* *
* * *
* * * *
* * * * *
```

Satır içi döngü yapıları (inline loops)

C dilinde satır içi döngüler, genellikle kısa ve basit döngü işlemlerini tek bir satırda yazmak için kullanılan bir tekniktir. Bu tür döngüler, kodun daha kompakt ve okunabilir olması amacıyla tercih edilir. En sık kullanılan satır içi döngü yapıları "for" ve "while" döngüleridir. Bu döngüler, kodun mantığını karmaşıklaştırmayacak kadar basit olduklarında, özellikle kısa ve tek bir işlem yapacaklarsa, bir satırda yazılabilirler.

Örneğin, bir dizi elemanını tek tek ekrana yazdırmak için for döngüsünü tek satırda kullanabiliriz:

```c
int dizi[] = {1, 2, 3, 4, 5};

for (int i = 0; i < 5; i++) printf("%d ", dizi[i]);
// Dizinin her elemanını yazdırır
```

"while" döngüsü de benzer şekilde tek bir işlem için satır içinde kullanılabilir. Örneğin, bir sayacın değeri belirli bir koşulu sağlayana kadar artması gerektiğinde, **"while"** döngüsünü kısa bir işlemle tek satırda yazabiliriz:

```c
int i = 0;

while (i < 5) printf("%d ", i++);
// i değeri 5'ten küçük olduğu sürece ekrana yazdırılır
```

Birden fazla işlem gerektiren döngüler veya karmaşık koşullar içeren döngüler satır içinde yazılmamalıdır. Özellikle uzun ve karmaşık işlemler tek satırda yazıldığında, kodun bakımını zorlaştırabilir ve hata yapma olasılığını artırabilir.

While döngüsü kullanımı

"while" döngüsü, **"for"** döngüsünden farklı olarak bir sayı dizisi veya liste üzerinden değil, bir koşulun sağlanıp sağlanmadığına bakarak çalışır. Koşul yanlış (False) olduğunda döngü sona erer. Bu nedenle, **"while"** döngüsü genellikle koşula dayalı işlemler için kullanılır.

Örnek 1: Basit Bir While Döngüsü

Bir koşul doğru olduğu sürece tekrar tekrar işlem yapılır. Örneğin, bir sayacı 1'den 5'e kadar arttıralım.

```
int sayi = 1;         // Sayaç değişkeni

while (sayi <= 5){    // i 5'ten küçük veya eşit olduğu sürece

    printf("%d\n", sayi); // i'nin güncel değerini ekrana yazdır

    sayi += 1;            // i'yi 1 artır
}
```

çıktısı

```
1
2
3
4
5
```

Bu örnekte, "while" döngüsü, "sayi <= 5" koşulu doğru olduğu sürece çalışır. Her döngüde "sayi" 1 arttırılır ve ekrana yazdırılır. "sayi" 6 olduğunda koşul yanlış olur ve döngü sona erer.

Örnek 2: Sonsuz Döngü

Eğer **"while"** döngüsündeki koşul her zaman doğru kalırsa, döngü asla bitmez ve bu duruma sonsuz döngü denir. Sonsuz döngüden çıkmak için döngü içinde belirli bir koşulla döngüyü sonlandırmak gerekebilir.

```
while (1){
    printf("Bu bir sonsuz döngüdür.");
    break;  // döngüyü durdurmak için kullanılır
}
```

While döngüsü kullanımı

Örnek 3: Kullanıcı Girişi ile While Döngüsü

"while" döngüsü, bir kullanıcı belirli bir koşulu sağladığında sonlanabilir. Örneğin, kullanıcı doğru bir şifre girdiğinde döngüyü sonlandıralım.

```
int sifre = 12345;
int giris = -1;
while (giris != sifre){
    printf("Lütfen şifrenizi girin: ");
    scanf("%d", &giris);
}
printf("Doğru şifreyi girdiniz!");
```

Bu örnekte, kullanıcı doğru şifreyi girene kadar her defasında şifre girilmesi istenir. Şifre doğru girildiğinde döngü biter ve "Doğru şifreyi girdiniz!" mesajı ekrana yazdırılır.

Örnek 4: While döngüsü ile liste toplamını hesaplama

```
int dizi[] = {1, 2, 3, 4, 5};
int toplam = 0, i = 0;
while (i <= 5){
    toplam += dizi[i];
    i += 1;
}
printf("Dizi toplamı: %d", toplam);
```

While döngüsü ve "continue" ve "break" ifadeleri

Döngüyü belirli bir koşul sağlandığında sonlandırmak için **"break"** komutu, döngüdeki belirli bir adımı atlayıp bir sonraki adıma geçmek için ise **"continue"** komutu kullanılır.

```
int sayi;
while (1) {
    printf("Bir sayı girin (çıkmak için 0 girin): ");
    scanf("%d", &sayi);
    if (sayi == 0){
        printf("Döngü sonlandırıldı.\n");
        break;
    }
    printf("Girdiğiniz sayı: %d\n", sayi);
}
```

```
Bir sayı girin (çıkmak için 0 girin): 5
Girdiğiniz sayı: 5
Bir sayı girin (çıkmak için 0 girin): 3
Girdiğiniz sayı: 3
Bir sayı girin (çıkmak için 0 girin): 0
Döngü sonlandırıldı.
```

çıktısı

Bu örnekte, kullanıcı sıfır "0" sayısını girdiğinde **"break"** komutu ile döngü sona erer.

Örneğin, 10'a kadar olan tek sayıları yazdıralım.

```
int sayi = 0;
while (sayi < 10){
    sayi += 1;
    if (sayi%2 == 0) continue;  # çift sayılar atlanır
    printf("%d", sayi);
}
```

çıktısı

```
1
3
5
7
9
```

Bu örnekte, **"sayi"** çift olduğunda **"continue"** komutu devreye girer ve o adım atlanır. Sadece tek sayılar ekrana yazdırılır.

İç-içe (nested) "while" döngü yapısı

"while" döngüsü de **"for"** döngüsüne benzer şekilde iç-içe kullanılabilir. "for" döngüsünde olduğu gibi, bir "while" döngüsünün içinde başka bir "while" döngüsü konulabilir. İç içe "while" döngüleri, özellikle birbiriyle bağlantılı tekrar eden işlemler için daha kullanışlıdır.

```
while (dış_koşul){
    //
    // dış döngü işlemleri (varsa)
    //
    while (iç_koşul){
        //
        // iç döngü işlemler
        //
    }
}
```

Burada öncelikle dıştaki döngü çalışır. Dış koşul doğru olduğu sürece bu döngü devam eder. Dış döngü her çalıştığında, iç döngü tekrar baştan başlar ve iç koşul doğru olduğu sürece çalışır.

```
int sayi = 1, tekrar;
while (sayi <= 5){
    tekrar = sayi;
    while (tekrar > 0){
        printf("%d ", sayi);
        tekrar -= 1;
    }
    printf("\n");
    sayi += 1;
}
printf("işlem tamamlandı...\n");
```

çıktısı

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

işlem tamamlandı...
```

İç-içe "while" döngü örneği

Bir listede verilen metin değişkenlerini (örneğin bir isim listesi) metindeki harf sayısı kadar yazdıran bir programı oluşturalım.

İlk adımda listedeki elemanları tek tek işleme sokmak için bir **"for-in"** döngüsü kullanılmalıdır. Daha sonra **"len"** uzunluk fonksiyonu ile harf sayısı elde edilmelidir. Son adımda **"while"** döngüsü ile harf sayısı azaltılarak eleman ekrana yazılmalıdır.

```
char kişiler[][10] = {"Ümit", "İlkay", "Kamuran", "Oya"};
int tekrar, i = 0;
while (i < 5){
    tekrar = strlen(kişiler[i]);    // string.h kullanılmalı
    while (tekrar > 0){
        printf("%s ", kişiler[i]);
        tekrar -= 1;
    }
    print("\n");
    i++;
}
printf("işlem tamamlandı...\n");
```

çıktısı

```
Ümit Ümit Ümit Ümit
İlkay İlkay İlkay İlkay İlkay
Kamuran Kamuran Kamuran Kamuran Kamuran Kamuran Kamuran
Oya Oya Oya
işlem tamamlandı...
```

"do-while" döngüsü kullanımı

"do-while" döngüsü, bir koşulun doğru olup olmadığını kontrol etmeden önce döngü gövdesini en az bir kez çalıştıran bir döngü yapısıdır. "do-while" döngüsünün en önemli özelliği, döngü gövdesinin mutlaka bir kere çalıştırılmasıdır; bu nedenle, döngünün en az bir kez çalışması gereken durumlarda tercih edilir.

Örneğin kullanıcıdan 1 ila 10 arası bir sayı girmesini isteyen ve sayı bu aralıkta verilinceye kadar sayı istemeye devam eden bir durumu dikkate alalım. Buna göre;

```c
int sayi;
do {
    printf("1 ile 10 arasinda bir sayi girin: ");
    scanf("%d", &sayi);
    if (sayi < 1 || sayi > 10) {
        printf("Geçersiz giriş! Lütfen 1 ile 10 arasında bir sayı girin.\n");
    }
} while (sayi < 1 || sayi > 10);
printf("Girilen geçerli sayi: %d\n", sayi);
```

çıktısı

```
1 ile 10 arasinda bir sayi girin: 78
Geçersiz giriş! Lütfen 1 ile 10 arasında bir sayı girin.
1 ile 10 arasinda bir sayi girin: 23
Geçersiz giriş! Lütfen 1 ile 10 arasında bir sayı girin.
1 ile 10 arasinda bir sayi girin: 5
Girilen geçerli sayi: 5
```

Örnek soru: 1

"Ben", "Sen", "O", "Biz" ve **"Onlar"** şahıs zamirleri için **"okuyor", "yazıyor"** ve **"yürüyor"** eylemlerini kullanan ve her bir şahıs zamirinin şahıs eklerini dikkate alarak fiil çekimlerini oluşturan bir programı oluşturalım.

İlk adımda şahıs zamirleri için bir liste oluşturulmalıdır. Daha sonra şahıs ekleri için diğer bir liste yapılmalıdır. Ardında eylemler belirlenmeli ve bunlar bir listeye konulmalıdır. Daha sonra zamir listesi **"for-in"** döngüsü kullanılarak sırayla işleme konulur. Ancak burada şahıs eklerinin doğru elde edilmesi için **"enumerate"** fonksiyonu kullanılmalıdır. Böylece her bir şahı zamirinin eki sırasına göre bulunur. Sonra eylem listesi için yine **"for-in"** döngüsü kullanılır ve **"print-format"** ikilisiyle sonuçlar birleştirilerek ekrana yazılır.

```
char zamir[][10] = {"Ben", "Sen", "O", "Biz", "Onlar"};
int adet = sizeof(zamir) / sizeof(zamir[0]);
char sahis_ek[][5] = {"um", "sun", "", "uz", "lar"};
char eylem[][10] = {"okuyor", "yazıyor", "yürüyor"};
int adeteyl = sizeof(eylem) / sizeof(eylem[0]);

for (int i = 0; i < adet; i++){

    for (int k = 0; k < adeteyl; k++){

        printf("%s %s%s\n", zamir[i], eylem[k], sahis_ek[i]);
    }
    printf("\n");
}
printf("işlem tamamlandı...");
```

çıktısı

```
Ben okuyorum.
Ben yazıyorum.
Ben yürüyorum.

Sen okuyorsun.
Sen yazıyorsun.
Sen yürüyorsun.

O okuyor.
O yazıyor.
O yürüyor.

Biz okuyoruz.
Biz yazıyoruz.
Biz yürüyoruz.

Onlar okuyorlar.
Onlar yazıyorlar.
Onlar yürüyorlar.

işlem tamamlandı...
```

Örnek soru: 2

Bilgisayarın 1 ile 100 arasında hafızasında tuttuğu bir sayıyı kullanıcının tahmin etmesini sağlayan bir program oluşturalım. Program, doğru sayıyı tahmin edene kadar kullanıcıdan tekrar bir sayı istemelidir.

Çözüm:

İlk olarak bilgisayarın tuttuğu sayı ayarlanmalıdır. Ardında bir sonsuz döngü içinde kullanıcıdan tahmin ettiği sayı tamsayı olarak alınmalıdır. Daha sonra bu sayı hafızadaki sayı ile karşılaştırılmalıdır. Eğer kullanıcının girdiği sayı hafızadaki sayı ile aynı ise bir "tebrik" mesajı yazılmalı, değilse ise "tekrar deneme" mesajı yazılmalıdır. Sonsuz döngüden çıkmak için ise tahminin doğru olmasından sonra "break" komutu verilerek döngüden çıkılmalıdır. Ayrıca kullanıcının tahmin sayısına bir sınırlama koyulabilir ve bilgisayarın rastgele bir sayı tutması da sağlanabilir.

```
// #include <stdlib.h>
int tutulan = 42, tahmin;          # tutulan = 1 + rand()%99;

do {

    printf("1 ila 100 arasındaki sayı tahminiz: ");

    scanf("%d", &tahmin);


    if (tahmin != tutulan) printf("Doğru değil, Tekrar deneyin\n");


} while (tutulan != tahmin);

printf("\nTebrikler! Sayıyı doğru tahmin ettiniz.");
```

```c
// #include <stdlib.h>
int tutulan = 42, tahmin;          # tutulan = 1 + rand()%99;

do {
    printf("1 ila 100 arasındaki sayı tahminiz (0:çıkış): ");
    scanf("%d", &tahmin);

    if (tahmin == 0) break;
    if (tahmin != tutulan) printf("Doğru değil, Tekrar deneyin\n");

} while (tutulan != tahmin);

if (tahmin != 0) printf("\nTebrikler! Sayıyı doğru tahmin ettiniz.");
```

Örnek soru: 3

Klasik **"taş", kağıt", "makas"** oyununu oluşturalım. Burada kişi bilgisayara karşı oynayacaktır. Bilgisayar bir seçim yapacak kişi uygun cevaplar vererek bilgisayarı yenmeye çalışacaktır.

Çözüm:

İlk adımda "taş", "kağıt", "makas" kelimeleri bir seçim yapılabilmesi için bir liste içine konmalıdır. Bilgisayarın bu listeden rastgele bir seçim yapması C kütüphanesinden "stdlib.h" başlık dosyasını yüklenmesi gerekir. Kullanıcıdan doğru cevap gelinceye kadar sürekli tahmin alınması için bir sonsuz döngü oluşturulur. "taş-makas", "kağıt-taş" ve "makas-kağıt" eşleşmesinde kişi kazanır aksi durumda bilgisayar kazanır.

```
int kull, bilg;
char *secim[] = {"taş", "kağıt", "makas"};

while (1){
    printf("Taş-Kağıt-Makas Oyununa Hoşgeldiniz!\n");
    printf("Seçiminizi yapın (1:taş, 2:kağıt, 3:makas): ");
    scanf("%d", &kull);

    srand(time(0));                          // #include <time.h>
    bilg = rand() % 3;                       // #include <stdlib.h>
    printf("Seçimler: %s - %s\n", secim[bilg], secim[kull]);

    if (kull == bilg) {
        printf("Beraberlik!\n");
    } else if ((kull == 1 && bilg == 3) ||
               (kull == 2 && bilg == 1) ||
               (kull == 3 && bilg == 2)) {
        printf("Kazandınız!\n");

    } else {printf("Kaybettiniz!\n");}

    printf("Devam edilsin mi? (e/h): ");
    scanf("%c", &tekrar);

    if (tekrar == 'h') break;
}
```

Çalışma soruları

Soru 1. 1'den 10' kadar sayıların karelerinin toplamanı hesaplayan programı yazınız.

Soru 2. 2 ila 100 arasında tam kare ve çift sayı olan sayıları bulan programı yazınız.
Not: Tam kare sayı kendi karekökü ile kalansız bölünen sayıdır. Karekök "math.h" başlık dosyası ve "sqrt()" ile kullanılır. Mesajı "... sayısı çift ve ... sayısının karesidir." şeklinde kullanınız.

Soru 3. Aşağıdaki şekli oluşturan programı yazınız.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```