Kontrol Akışı ve Döngüler

Table of contents

Bölüm 3: Kontrol Akışı ve Döngüler

Bir programın akışı, her zaman yukarıdan aşağıya doğru ilerlemez. Bazen belirli koşullara bağlı olarak farklı kod bloklarını çalıştırmamız veya belirli bir kod bloğunu tekrar tekrar çalıştırmamız gerekebilir. İşte bu noktada kontrol akışı ifadeleri devreye girer. Kontrol akışı ifadeleri, programın akışını yönlendirmemizi ve belirli koşullara göre farklı kod bloklarını çalıştırmamızı sağlar. Bu bölümde, C dilinde kullanılan temel kontrol akışı ifadelerini, `if-else` yapısını, `switch-case` yapısını ve döngüleri (`for`, `while`, `do-while`) inceleyeceğiz.

3.1 `if-else` Yapısı: Karar Verme Mekanizması

`if-else` yapısı, belirli bir koşul doğruysa bir kod bloğunu, yanlışsa başka bir kod bloğunu çalıştırmamızı sağlar. Koşul, mantıksal bir ifadedir ve sonucu `doğru` veya `yanlış` olabilir.

`if-else` Yapısı Sözdizimi:

```
if (koşul) {
  // Koşul doğruysa çalışacak kod bloğu
} else {
  // Koşul yanlışsa çalışacak kod bloğu
}
```

Örnek: Sayının Pozitif, Negatif veya Sıfır Olduğunu Kontrol Etme

```
int sayi;

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

if (sayi > 0) {
  printf("Sayı pozitiftir.\n");
} else if (sayi < 0) {
  printf("Sayı negatiftir.\n");
} else {
  printf("Sayı sıfırdır.\n");
}
```

Daha Fazla `if-else` Örneği:

- **Kullanıcının girdiği sayının çift mi tek mi olduğunu kontrol etme:**

```
int sayi;

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

if (sayi % 2 == 0) {
  printf("Sayı çifttir.\n");
} else {
  printf("Sayı tektir.\n");
}
```

- **Kullanıcının girdiği iki sayıdan büyük olanı bulma:**

```
int sayi1, sayi2;

printf("Birinci sayıyı girin: ");
scanf("%d", &sayi1);
```

```
printf("İkinci sayıyı girin: ");
scanf("%d", &sayi2);

if (sayi1 > sayi2) {
  printf("Birinci sayı daha büyüktür.\n");
} else if (sayi2 > sayi1) {
  printf("İkinci sayı daha büyüktür.\n");
} else {
  printf("Sayılar eşittir.\n");
}
```

- **Kullanıcının girdiği notlara göre harf notu hesaplama:**

```
int not;

printf("Notunuzu girin: ");
scanf("%d", &not);

if (not >= 90) {
  printf("Harf notunuz: AA\n");
} else if (not >= 80) {
  printf("Harf notunuz: BA\n");
} else if (not >= 70) {
  printf("Harf notunuz: BB\n");
} else if (not >= 60) {
  printf("Harf notunuz: CB\n");
} else if (not >= 50) {
  printf("Harf notunuz: CC\n");
} else {
  printf("Harf notunuz: FF\n");
}
```

3.2 Ternary Operatörü: Koşullu İfadeleri Kısaltmak

Ternary operatörü (`?:`), `if-else` yapısının kısa bir alternatifidir ve tek bir satırda koşullu ifadeler yazmamızı sağlar.

Ternary Operatörü Sözdizimi:

```
koşul ? ifade1 : ifade2
```

- **`koşul`:** Değerlendirilmek üzere bir mantıksal ifade.

- **ifade1:** koşul doğru ise değerlendirilecek ifade.
- **ifade2:** koşul yanlış ise değerlendirilecek ifade.

Örnek: En Büyük Sayıyı Bulma

```
int sayi1 = 10;
int sayi2 = 20;
int enBuyuk = (sayi1 > sayi2) ? sayi1 : sayi2; // Ternary operatörü

printf("En büyük sayı: %d\n", enBuyuk); // Çıktı: En büyük sayı: 20
```

Bu örnekte, `sayi1 > sayi2` koşulu doğru ise `enBuyuk` değişkenine `sayi1` değeri, yanlış ise `sayi2` değeri atanır.

Ternary Operatörünün Avantajları:

- **Kısa ve Öz:** `if-else` yapısına göre daha kısa ve okunaklı olabilir.
- **Tek Satırda Koşullu Atama:** Değişkenlere koşullu olarak değer atamak için pratik bir yöntem.

Örnek: Sayının Tek mi Çift mi Olduğunu Kontrol Etme

```
int sayi = 7;
char *sonuc = (sayi % 2 == 0) ? "Çift" : "Tek";

printf("%d sayısı %s'tir.\n", sayi, sonuc); // Çıktı: 7 sayısı Tek'tir.
```

3.3 `switch-case` Yapısı: Çoklu Seçenekler Arasında Gezinme

`switch-case` yapısı, bir değişkenin değerine bağlı olarak farklı kod bloklarını çalıştırmamızı sağlar. Değişkenin değeri, `case` etiketleri ile karşılaştırılır ve eşleşen `case` bloğu çalıştırılır.

`switch-case` Yapısı Sözdizimi:

```
switch (değişken) {
  case değer1:
    // Değişkenin değeri değer1 ise çalışacak kod bloğu
    break;
  case değer2:
    // Değişkenin değeri değer2 ise çalışacak kod bloğu
    break;
  // ...
  default:
```

```
    // Hiçbir case etiketi eşleşmezse çalışacak kod bloğu
}
```

Örnek: Haftanın Gününü Yazdırma

```
int gun;

printf("Haftanın gününü girin (1-7): ");
scanf("%d", &gun);

switch (gun) {
  case 1:
    printf("Pazartesi\n");
    break;
  case 2:
    printf("Salı\n");
    break;
  case 3:
    printf("Çarşamba\n");
    break;
  case 4:
    printf("Perşembe\n");
    break;
  case 5:
    printf("Cuma\n");
    break;
  case 6:
    printf("Cumartesi\n");
    break;
  case 7:
    printf("Pazar\n");
    break;
  default:
    printf("Geçersiz gün!\n");
}
```

Daha Fazla `switch-case` Örneği:

- **Hesap Makinesi:**

```c
char operator;
float sayi1, sayi2, sonuc;

printf("İki sayı girin: ");
scanf("%f %f", &sayi1, &sayi2);

printf("Operatörü girin (+, -, *, /): ");
scanf(" %c", &operator);

switch (operator) {
  case '+':
    sonuc = sayi1 + sayi2;
    printf("Sonuç: %.2f\n", sonuc);
    break;
  case '-':
    sonuc = sayi1 - sayi2;
    printf("Sonuç: %.2f\n", sonuc);
    break;
  case '*':
    sonuc = sayi1 * sayi2;
    printf("Sonuç: %.2f\n", sonuc);
    break;
  case '/':
    if (sayi2 == 0) {
      printf("Sıfıra bölme hatası!\n");
    } else {
      sonuc = sayi1 / sayi2;
      printf("Sonuç: %.2f\n", sonuc);
    }
    break;
  default:
    printf("Geçersiz operatör!\n");
}
```

- **Ayları Yazdırma:**

```c
int ay;

printf("Ay numarasını girin (1-12): ");
scanf("%d", &ay);

switch (ay) {
```

```c
  case 1:
    printf("Ocak\n");
    break;
  case 2:
    printf("Şubat\n");
    break;
  // ... diğer aylar ...
  default:
    printf("Geçersiz ay!\n");
}
```

3.3 Döngüler: Tekrarlayan İşlemler için Pratik Çözümler

Döngüler, belirli bir kod bloğunu tekrar tekrar çalıştırmamızı sağlar. C dilinde üç temel döngü türü vardır: `for`, `while` ve `do-while`.

3.3.1 `for` Döngüsü: Belirli Sayıda Tekrar

`for` döngüsü, belirli bir sayıda tekrar yapmak için kullanılır. Döngü değişkeninin başlangıç değeri, bitiş koşulu ve artış miktarı belirtilir.

`for` Döngüsü Sözdizimi:

```c
for (başlangıç_değeri; bitiş_koşulu; artış_miktarı) {
  // Tekrarlanacak kod bloğu
}
```

`i++` ve `i--` Operatörleri

- `i++` operatörü, `i` değişkeninin değerini her adımda 1 artırır.
- `i--` operatörü, `i` değişkeninin değerini her adımda 1 azaltır.

Örnek: 1'den 10'a Kadar Sayıları Yazdırma

```c
for (int i = 1; i <= 10; i++) {
  printf("%d ", i);
}
printf("\n");
```

Daha Fazla `for` Döngüsü Örneği:

- **Çift Sayıları Yazdırma:**

```
for (int i = 2; i <= 20; i += 2) {
  printf("%d ", i);
}
printf("\n");
```

- **Geri Sayım Yapma:**

```
for (int i = 10; i >= 1; i--) {
  printf("%d ", i);
}
printf("Kalkış!\n");
```

- **Yıldızlardan Üçgen Oluşturma:**

```
for (int i = 1; i <= 5; i++) {
  for (int j = 1; j <= i; j++) {
    printf("*");
  }
  printf("\n");
}
```

3.3.1.1 İç İçe `for` Döngüleri

`for` döngülerini iç içe kullanarak daha karmaşık işlemler gerçekleştirebiliriz. İç içe döngülerde, dış döngünün her bir iterasyonu için iç döngü tamamen çalışır.

Örnek: Çarpım Tablosu

```
#include <stdio.h>

int main() {
  for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
      printf("%d x %d = %d\n", i, j, i * j);
    }
    printf("\n");
  }
  return 0;
}
```

Bu program, 1'den 10'a kadar olan sayılar için çarpım tablosunu oluşturur. Dış döngü (`i` değişkeni ile kontrol edilir) her satırı temsil ederken, iç döngü (`j` değişkeni ile kontrol edilir) her satırdaki çarpım işlemlerini gerçekleştirir. `i * j` işleminin sonucu, her bir çarpımın sonucunu verir. `printf("\n");` satırı ise her satırdan sonra bir boş satır ekler, böylece çarpım tablosu daha okunaklı olur. ### 3.3.2 `while` Döngüsü: Koşul Doğru Olduğu Sürece Tekrar

`while` döngüsü, belirli bir koşul doğru olduğu sürece tekrar yapar. Döngüye girmeden önce koşul kontrol edilir.

`while` Döngüsü Sözdizimi:

```
while (koşul) {
  // Tekrarlanacak kod bloğu
}
```

Örnek: Kullanıcı 0 Girinceye Kadar Sayıları Toplama

```
int sayi, toplam = 0;

printf("Bir sayı girin (0 girmek döngüyü sonlandırır): ");
scanf("%d", &sayi);

while (sayi != 0) {
  toplam += sayi;
  printf("Bir sayı girin (0 girmek döngüyü sonlandırır): ");
  scanf("%d", &sayi);
}

printf("Toplam: %d\n", toplam);
```

Daha Fazla `while` Döngüsü Örneği:

- **Kullanıcının Girdiği Sayıların Ortalamasını Hesaplama:**

```
int sayi, sayac = 0, toplam = 0;
float ortalama;

printf("Bir sayı girin (0 girmek döngüyü sonlandırır): ");
scanf("%d", &sayi);

while (sayi != 0) {
  toplam += sayi;
  sayac++;
```

```
  printf("Bir sayı girin (0 girmek döngüyü sonlandırır): ");
  scanf("%d", &sayi);
}

if (sayac > 0) {
  ortalama = (float)toplam / sayac;
  printf("Ortalama: %.2f\n", ortalama);
} else {
  printf("Hiç sayı girilmedi.\n");
}
```

- **Rastgele Sayı Üretme (Belirli Bir Koşul Sağlanana Kadar):**

```
#include <stdlib.h>
#include <time.h>

int main() {
  int rastgeleSayi;
  srand(time(NULL)); // Rastgele sayı üreteci başlatılıyor

  while (1) {
    rastgeleSayi = rand() % 100; // 0 ile 99 arasında rastgele sayı üretir
    printf("Rastgele Sayı: %d\n", rastgeleSayi);
    if (rastgeleSayi == 50) {
      printf("50 bulundu! Döngü sonlandırılıyor.\n");
      break;
    }
  }

  return 0;
}
```

3.3.3 `do-while` Döngüsü: En Az Bir Kere Tekrar

`do-while` döngüsü, `while` döngüsüne benzer, ancak koşul kontrolü döngü bloğu çalıştırıldıktan sonra yapılır. Bu nedenle, döngü bloğu en az bir kere çalıştırılır.

`do-while` Döngüsü Sözdizimi:

```
do {
  // Tekrarlanacak kod bloğu
} while (koşul);
```

Örnek: Kullanıcı Geçerli Bir Sayı Girinceye Kadar Tekrar Sorma

```
int sayi;

do {
  printf("1 ile 10 arasında bir sayı girin: ");
  scanf("%d", &sayi);
} while (sayi < 1 || sayi > 10);

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

Daha Fazla `do-while` Döngüsü Örneği:

- **Kullanıcıdan Şifre İsteme (Doğru Girilene Kadar):**

```
char sifre[20];

do {
  printf("Şifreyi girin: ");
  scanf("%s", sifre);

  if (strcmp(sifre, "gizli_sifre") != 0) {
    printf("Şifre yanlış! Tekrar deneyin.\n");
  }
} while (strcmp(sifre, "gizli_sifre") != 0);

printf("Şifre doğru.\n");
```

- **Menü Seçeneklerini Sunma (Kullanıcı Çıkış Yapana Kadar):**

```
int secim;

do {
  printf("Menü:\n");
  printf("1. Seçenek 1\n");
  printf("2. Seçenek 2\n");
  printf("3. Çıkış\n");
  printf("Seçiminizi yapın: ");
  scanf("%d", &secim);

  switch (secim) {
    case 1:
```

```
      // Seçenek 1 işlemleri
      printf("Seçenek 1 seçildi.\n");
      break;
    case 2:
      // Seçenek 2 işlemleri
      printf("Seçenek 2 seçildi.\n");
      break;
    case 3:
      printf("Çıkış yapılıyor...\n");
      break;
    default:
      printf("Geçersiz seçim!\n");
  }

} while (secim != 3);
```

3.3.4 `break` İfadesi: Döngüyü Sonlandırmak

`break` ifadesi, bir döngüyü (`for`, `while` veya `do-while`) hemen sonlandırmak için kullanılır. `break` ifadesine ulaşıldığında, döngü sona erer ve program akışı döngüden sonraki ifadeye geçer.

Örnek: `break` İfadesi ile Döngüyü Sonlandırma

```
for (int i = 1; i <= 10; i++) {
  if (i == 5) {
    break; // i değeri 5 olduğunda döngü sonlandırılır
  }
  printf("%d ", i);
}
printf("\n"); // Çıktı: 1 2 3 4
```

Bu örnekte, `for` döngüsü normalde 1'den 10'a kadar olan sayıları yazdırır. Ancak, `i == 5` koşulu sağlandığında, `break` ifadesi çalışır ve döngü sona erer. Bu yüzden çıktı sadece 1, 2, 3 ve 4 olur.

3.3.5 `continue` İfadesi: Döngünün Bir Sonraki Adımına Geçmek

`continue` ifadesi, döngünün **mevcut iterasyonunu** sonlandırır ve döngünün bir sonraki adımına geçer. `continue` ifadesine ulaşıldığında, döngü bloğunun geri kalanı atlanır ve döngü koşulunun kontrolüne geri dönülür.

Örnek: `continue` İfadesi ile Tek Sayıları Atlama

```
for (int i = 1; i <= 10; i++) {
  if (i % 2 == 0) {
    continue; // i değeri çift ise, bu iterasyon atlanır
  }
  printf("%d ", i);
}
printf("\n"); // Çıktı: 1 3 5 7 9
```

Bu örnekte, `for` döngüsü 1'den 10'a kadar olan sayıları yazdırır. Ancak, `i % 2 == 0` koşulu sağlandığında (yani `i` çift olduğunda), `continue` ifadesi çalışır ve döngü bloğunun geri kalanı atlanır. Bu yüzden çıktı sadece tek sayıları (1, 3, 5, 7, 9) içerir.

3.4 Sonuç

Bu bölümde, programın akışını kontrol etmek için kullanılan temel kontrol akışı ifadelerini öğrendik. `if-else` yapısı ile karar vermeyi, `switch-case` yapısı ile çoklu seçenekler arasında gezinmeyi, döngüler ile tekrarlayan işlemleri nasıl yapacağımızı ve `break` ve `continue` ifadeleri ile döngü akışını nasıl kontrol edeceğimizi gördük.