Değişkenler, Veri Tipleri ve Operatörler

Table of contents

Bölüm 2: Temel Veri Tipleri ve Operatörler - C Dilinin Alfabesi

Bilgisayar programları, verileri işleyerek çalışır. Bu veriler, sayılar, metinler, mantıksal değerler gibi farklı tiplerde olabilir. Tıpkı bir dilin alfabesi gibi, C programlama dilinin de temel yapı taşları olan veri tipleri ve operatörleri vardır. Bu bölümde, C dilinin alfabesini öğrenecek ve verileri nasıl saklayacağımızı, işleyeceğimizi ve kullanacağımızı göreceğiz.

2.1 Değişkenler: Verileri Saklama Kutuları

Değişkenler, verileri depolamak için kullanılan isimlendirilmiş bellek bölgeleridir. Bir değişken, tıpkı bir kutu gibi, içinde belirli bir türde veri saklayabilir. C dilinde, değişkenleri kullanmadan önce tanımlamamız gerekir. Değişken tanımlama işlemi, değişkenin adını ve veri tipini belirterek yapılır.

```
veri_tipi değişken_adı;
```

Değişken Tanımlama Örneği:

```c
int yas;          // Tam sayı (integer) tipinde bir değişken
float boy;        // Ondalıklı sayı (floating-point) tipinde bir değişken
char cinsiyet;    // Tek karakter (character) tipinde bir değişken
```

Bellek Diyagramı

```
+-------+-------+-------+-------+
| Adres | 0x100 | 0x104 | 0x108 |  ...
+-------+-------+-------+-------+
| Değer |   10  |  20.5 |   'A'  |  ...
+-------+-------+-------+-------+
     ^       ^       ^
     |       |       |
     int x  float y char ch
```

Bu diyagramda, sadece temel veri tipleri (int, float, char) ve bunların bellekte nasıl saklandığı gösteriliyor. Her değişken, bellekte bir adres aralığına sahip ve bu adres aralığında değişkenin değeri saklanıyor.

Değişken İsimlendirme Kuralları:

- Değişken isimleri harf, rakam ve alt çizgi (_) karakterlerinden oluşabilir.
- Değişken isimleri bir rakamla başlayamaz.
- Değişken isimleri büyük/küçük harf duyarlıdır (`isim` ile `Isim` farklı değişkenlerdir).
- C dilinde rezerve edilmiş kelimeler (int, float, char vb.) olamaz.

Daha Fazla Değişken Tanımlama Örneği:

```c
int sayi = 10;
float pi = 3.14159;
char harf = 'A';
char isim[] = "Ahmet";
```

2.2 Temel Veri Tipleri: Verilerin Çeşitliliği

C dilinde, farklı türde verileri saklamak için farklı veri tipleri kullanılır. En temel veri tipleri şunlardır:

- **`int` (Tam Sayı):** Tam sayıları (pozitif, negatif veya sıfır) saklamak için kullanılır. Örneğin: 10, -5, 0
- **`float` (Ondalıklı Sayı):** Ondalıklı sayıları saklamak için kullanılır. Örneğin: 3.14, -2.5, 0.0

- **`char` (Karakter):** Tek bir karakteri saklamak için kullanılır. Örneğin: 'A', 'b', '5', '?'

Veri Tipi Seçimi:

Hangi veri tipini kullanacağımız, saklamak istediğimiz veriye bağlıdır. Örneğin, bir kişinin yaşını saklamak için `int` veri tipini, boyunu saklamak için `float` veri tipini, cinsiyetini saklamak için ise `char` veri tipini kullanabiliriz.

2.3 Değişken Tanımlama ve İlklendirme

C dilinde, bir değişkeni kullanmadan önce **tanımlamak** gerekir. Tanımlama işlemi, değişkene bir isim ve veri tipi vermektir. Bu, bellekte o değişken için yer ayırır, ancak henüz bir değer atamaz. **İlklendirme** ise, tanımlanan değişkene ilk değeri vermektir.

Değişken Tanımlama Sözdizimi:

```
veri_tipi değişken_adı;
```

Değişken İlklendirme Sözdizimi:

```
değişken_adı = değer;
```

Tanımlama ve İlklendirme Birlikte:

Değişkenleri tanımlarken aynı zamanda ilk değerlerini de atayabiliriz.

```
veri_tipi değişken_adı = değer;
```

Örnekler:

```
int yas;              // Tanımlama: int tipinde "yas" adında bir değişken tanımlandı.
yas = 25;            // İlklendirme: "yas" değişkenine 25 değeri atandı.

float boy = 1.75;    // Tanımlama ve ilklendirme birlikte: float tipinde "boy" adında bir değ

char cinsiyet = 'E';// Tanımlama ve ilklendirme birlikte: char tipinde "cinsiyet" adında bir

int sayi1, sayi2 = 10; // "sayi1" tanımlandı, "sayi2" tanımlandı ve 10 değeri atandı.
```

> **Dikkat**
>
> Tanımlanmadan ilklendirilen değişkenler hata verir. Her değişken öncelikle tanımlanmalıdır.

> **İpucu**
>
> İlklendirilmeden kullanılan değişkenlerin değeri, bellekten gelen rastgele bir değerdir. Bu, beklenmedik sonuçlara yol açabilir. Bu yüzden, değişkenleri kullanmadan önce ilklendirmek iyi bir programlama uygulamasıdır.

2.4 Operatörler: Verileri İşleme Araçları

Operatörler, veriler üzerinde işlemler yapmak için kullanılan sembollerdir. C dilinde, farklı türde operatörler vardır:

- **Aritmetik Operatörler:** Matematiksel işlemler yapmak için kullanılır.
  - `+` (Toplama)
  - `-` (Çıkarma)
  - `*` (Çarpma)
  - `/` (Bölme)
  - `%` (Mod alma - Kalanı bulma)
- **Atama Operatörü:** Bir değişkene değer atamak için kullanılır.
  - `=` (Atama)
- **Karşılaştırma Operatörleri:** İki değeri karşılaştırmak için kullanılır.
  - `==` (Eşittir)
  - `!=` (Eşit Değildir)
  - `>` (Büyüktür)
  - `<` (Küçüktür)
  - `>=` (Büyük Eşittir)
  - `<=` (Küçük Eşittir)

  Karşılaştırma operatörü sonucu 1 ya da 0 şeklinde yanıt verir. 1 doğru(true) 0 yanlış(false).
- **Mantıksal Operatörler:** Mantıksal işlemler yapmak için kullanılır.
  - `&&` (VE)
  - `||` (VEYA)

– ! (DEĞİL)

Operatör Önceliği:

C dilinde, operatörlerin bir öncelik sırası vardır. Örneğin, çarpma ve bölme işlemleri, toplama ve çıkarma işlemlerinden önce yapılır. Parantez kullanarak işlem sırasını değiştirebiliriz.

Örnek:

```
int a = 10;
int b = 5;
int c = a + b * 2;   // c = 20 (Önce çarpma yapılır)
int d = (a + b) * 2;  // d = 30 (Önce parantez içi yapılır)
```

Daha Fazla Operatör Örneği:

```
int x = 5;
int y = 10;

// Aritmetik Operatörler
int toplam = x + y;  // toplam = 15
int fark = x - y;    // fark = -5
int carpim = x * y;  // carpim = 50
int bolum = y / x;   // bolum = 2
int kalan = y % x;   // kalan = 0

// Atama Operatörleri
x += 5;  // x = x + 5  (x artık 10)
y -= 2;  // y = y - 2  (y artık 8)

// Karşılaştırma Operatörleri
int esitMi = x == y;  // esitMi = 0
int farkliMi = x != y; // farkliMi = 1

// Mantıksal Operatörler
int kosul1 = x > 5 && y < 10;  // kosul1 = 0
int kosul2 = x > 5 || y <= 10;  // kosul2 = 1
```

2.5 `printf` Fonksiyonu ve Format Belirteçleri: Verileri Ekrana Yazdırmak

C dilinde, `printf` fonksiyonu, verileri biçimlendirilmiş bir şekilde ekrana yazdırmak için kullanılır. `printf` fonksiyonu, bir **format string**'i ve bu string içindeki **format belirteçleri**'ni kullanarak verileri biçimlendirir.

Format Belirteçleri:

Format belirteçleri, `%` sembolü ile başlar ve bir harf ile biter. Her belirteç, farklı bir veri tipini temsil eder.

Önemli Belirteçler:

Belirteç	Veri Tipi	Açıklama
`%d`	`int`	Tam sayı değerlerini yazdırır.
`%f`	`float`	Ondalıklı sayı değerlerini yazdırır.
`%c`	`char`	Tek bir karakteri yazdırır.
`%s`	`char[]` (string)	Bir karakter dizisini (string) yazdırır.
`%%`		`%` sembolünü yazdırır.
`\n`		Yeni satır karakteri.
`\t`		Tab karakteri (yatay sekme).

Örnekler:

```c
int sayi = 10;
float ondalikSayi = 3.14159;
char karakter = 'A';
char metin[] = "Merhaba Dünya!";

printf("Sayı: %d\n", sayi);
printf("Ondalık Sayı (2 basamaklı): %.2f\n", ondalikSayi);
printf("Karakter: %c\n", karakter);
printf("Metin: %s\n", metin);
printf("Yüzde işareti: %%\n");
printf("Yeni\nsatır\n");
printf("İsim:\tAhmet\n");
```

Çıktı:

```
Sayı: 10
Ondalık Sayı (2 basamaklı): 3.14
Karakter: A
Metin: Merhaba Dünya!
Yüzde işareti: %
Yeni
satır
İsim:    Ahmet
```

Açıklama:

- `%d`, `%f`, `%c` ve `%s` belirteçleri, sırasıyla `sayi`, `ondalikSayi`, `karakter` ve `metin` değişkenlerinin değerlerini yazdırır.
- `%.2f`, ondalık sayının virgülden sonra iki basamağını yazdırır.
- `%%`, `%` sembolünü yazdırır.
- `\n`, yeni satır karakteri, çıktıyı bir sonraki satıra taşır.
- `\t`, tab karakteri, çıktıda yatay bir sekme oluşturur.

2.6 `scanf` Fonksiyonu: Verileri Kullanıcıdan Almak

`scanf` fonksiyonu, kullanıcıdan veri almak için kullanılır. `printf` fonksiyonu gibi, `scanf` da format belirteçleri kullanır. `scanf` fonksiyonunda, değişkenlerin değerlerini okumak için değişkenlerin bellek adreslerini belirtmemiz gerekir. Bunun için, değişken adının önüne `&` (adres) operatörünü koyarız.

`scanf` Fonksiyonu Sözdizimi:

```
scanf("format_string", &değişken1, &değişken2, ...);
```

Örnekler:

```
int sayi;
float ondalikSayi;
char karakter;

printf("Bir tam sayı girin: ");
scanf("%d", &sayi); // &sayi: sayi değişkeninin bellek adresi
printf("Girdiğiniz sayı: %d\n", sayi);

printf("Bir ondalık sayı girin: ");
scanf("%f", &ondalikSayi);
printf("Girdiğiniz ondalık sayı: %f\n", ondalikSayi);

printf("Bir karakter girin: ");
scanf(" %c", &karakter); // Boşluk, önceki girdiden kalan yeni satır karakterini atlar
printf("Girdiğiniz karakter: %c\n", karakter);
```

Açıklama:

- `%d`, `%f` ve `%c` format belirteçleri, sırasıyla tam sayı, ondalık sayı ve karakter değerlerini okumak için kullanılır.
- `&` operatörü, değişkenin bellek adresini verir. `scanf` fonksiyonu, bu adrese okuduğu değeri yazar.

2.7 `const` Niteleyicisi: Değişmez Değerler

C dilinde, `const` niteleyicisi, bir değişkenin değerinin programın çalışması sırasında değiştirilemez olduğunu belirtmek için kullanılır. `const` ile tanımlanan değişkenlere **sabit** (constant) denir. Sabitler, programın başlangıcında bir kez değer atanır ve bu değer daha sonra değiştirilemez.

`const` Kullanımının Avantajları:

- **Hataların Önlenmesi:** Değişkenin yanlışlıkla değiştirilmesini önleyerek, programda oluşabilecek hataları azaltır.
- **Kodun Okunabilirliğini Artırır:** Değişkenin sabit bir değer olduğunu belirterek, kodun daha anlaşılır olmasını sağlar.
- **Derleyici Optimizasyonu:** Derleyici, `const` niteleyicisi ile tanımlanan değişkenlerin değerinin değişmeyeceğini bildiği için, kodu daha verimli bir şekilde optimize edebilir.

`const` Sözdizimi:

```
const veri_tipi değişken_adı = değer;
```

Örnekler:

```
const float PI = 3.14159;  // PI sabitinin değeri 3.14159 olarak tanımlanır
const int MAX_DEGER = 100; // MAX_DEGER sabitinin değeri 100 olarak tanımlanır

int main() {
  // PI = 3.14;  // Hata! const değişkenin değeri değiştirilemez
  // MAX_DEGER = 200; // Hata! const değişkenin değeri değiştirilemez

  printf("PI: %.5f\n", PI);
  printf("MAX_DEGER: %d\n", MAX_DEGER);

  return 0;
}
```

`const` Kullanım Alanları:

- **Matematiksel Sabitler:** PI, Euler sayısı (e) gibi matematiksel sabitleri tanımlamak için.
- **Fiziksel Sabitler:** Işık hızı, yer çekimi ivmesi gibi fiziksel sabitleri tanımlamak için.
- **Program Parametreleri:** Programın çalışma şeklini etkileyen, ancak programın çalışması sırasında değiştirilmemesi gereken değerleri tanımlamak için.
- **Dizi Boyutları:** Dizilerin boyutlarını sabit olarak tanımlamak için.

Not: `const` niteleyicisi, işaretçi (pointer) değişkenleri ile kullanıldığında farklı anlamlar ifade edebilir. Bu konuyu, ilerleyen bölümlerde daha detaylı olarak ele alacağız.

Bu bölümde, `printf` fonksiyonu ve format belirteçlerini kullanarak verileri nasıl biçimlendirilmiş bir şekilde ekrana yazdırabileceğinizi öğrendiniz. Ayrıca, `const` niteleyicisini kullanarak değişmez değerleri nasıl tanımlayabileceğinizi gördünüz. Bu bilgiler, C programlamada temel yapı taşlarını anlamak ve kullanmak için önemlidir.

2.8 Yorum Satırları: Kodunuzu Açıklamak

C dilinde, **yorum satırları**, kodun okunabilirliğini ve anlaşılırlığını artırmak için kullanılan, derleyici tarafından yok sayılan metinlerdir. Yorum satırları, kodun belirli bölümlerini açıklamak, kodun nasıl çalıştığını belgelemek veya geçici olarak devre dışı bırakmak için kullanılabilir.

C dilinde iki tür yorum satırı vardır:

- **Tek Satırlık Yorumlar:** `//` karakterleri ile başlar ve satır sonuna kadar devam eder.
- **Çok Satırlık Yorumlar:** `/*` karakterleri ile başlar ve `*/` karakterleri ile biter. Bu tür yorumlar birden fazla satıra yayılabilir.

Örnekler:

```
// Bu tek satırlık bir yorumdur.

/*
Bu çok satırlık bir yorumdur.
Birden fazla satıra yayılabilir.
*/

int sayi = 10;  // Bu yorum, sayi değişkenini açıklar.

/*
Bu kod bloğu, kullanıcıdan iki sayı alır ve toplamlarını yazdırır.
Ancak, şu anda devre dışı bırakılmıştır.
*/
/*
int sayi1, sayi2, toplam;
printf("İki sayı girin: ");
scanf("%d %d", &sayi1, &sayi2);
toplam = sayi1 + sayi2;
printf("Toplam: %d\n", toplam);
*/
```

İyi Programlama Uygulamaları:

- **Anlamlı Yorumlar:** Yorumlarınız açık, öz ve anlaşılır olmalıdır.
- **Gereksiz Yorumlardan Kaçının:** Açık ve anlaşılır bir kod, genellikle çok fazla yoruma ihtiyaç duymaz.
- **Kodu Belgelemek İçin Yorum Kullanın:** Fonksiyonların ne yaptığını, parametrelerinin ne olduğunu ve dönüş değerlerinin ne olduğunu açıklamak için yorumlar kullanın.
- **Geçici Olarak Kodu Devre Dışı Bırakmak İçin Yorum Kullanın:** Hata ayıklama veya test sırasında, kodun belirli bölümlerini geçici olarak devre dışı bırakmak için yorumlar kullanabilirsiniz.