Standart Kütüphane Fonksiyonları

Table of contents

Bölüm 4: Standart Kütüphane Fonksiyonları - C'nin Hazır Araçları **1**
4.1 Başlık Dosyaları: Kütüphane Fonksiyonlarına Erişim 1
4.2 Yaygın Kullanılan Standart Kütüphane Fonksiyonları 2
4.2.1 `<stdio.h>`: Girdi/Çıktı İşlemleri 2
4.2.2 `<math.h>`: Matematiksel İşlemler 4
4.2.3 `<stdlib.h>`: Genel Amaçlı Fonksiyonlar 5
4.2.4 `<time.h>`: Zaman ve Tarih İşlemleri 7
4.3 Rastgele Sayı Üretimi 8
Örnek: Zar Atma Simülasyonu 8
4.4 Alıştırmalar 8
4.5 Sonuç 9

Bölüm 4: Standart Kütüphane Fonksiyonları - C'nin Hazır Araçları

C programlama dili, **standart kütüphane** adı verilen bir dizi hazır fonksiyon koleksiyonuyla gelir. Bu fonksiyonlar, programcıların sık kullanılan işlemleri kolayca gerçekleştirmesini sağlar. Girdi/çıktı işlemleri, matematiksel hesaplamalar, string manipülasyonu, bellek yönetimi, zaman ve tarih işlemleri gibi birçok farklı amaç için fonksiyonlar mevcuttur. Bu bölümde, C'nin standart kütüphanesindeki bazı önemli fonksiyonları ve bunları nasıl kullanabileceğinizi öğreneceksiniz.

4.1 Başlık Dosyaları: Kütüphane Fonksiyonlarına Erişim

Standart kütüphane fonksiyonları, **başlık dosyaları** (`.h` uzantılı dosyalar) içinde tanımlanmıştır. Bir fonksiyonu kullanmadan önce, ilgili başlık dosyasını `#include` direktifi ile kaynak koda eklemeniz gerekir.

Örnek:

```c
#include <stdio.h> // printf ve scanf fonksiyonları için gerekli

int main() {
  printf("Merhaba Dünya!\n");
  return 0;
}
```

Bu örnekte, `#include <stdio.h>` direktifi, standart girdi/çıktı kütüphanesinin (`stdio.h`) başlık dosyasını programa ekler. Bu başlık dosyası, `printf` fonksiyonunun tanımını içerir, bu yüzden programda `printf` fonksiyonunu kullanabiliriz.

4.2 Yaygın Kullanılan Standart Kütüphane Fonksiyonları

Şimdi, farklı başlık dosyalarında bulunan bazı yaygın kullanılan standart kütüphane fonksiyonlarına ve bunların nasıl kullanılacağına dair örnekler inceleyelim.

4.2.1 `<stdio.h>`: Girdi/Çıktı İşlemleri

`<stdio.h>` başlık dosyası, standart girdi/çıktı işlemleri için fonksiyonlar sağlar. En sık kullanılan fonksiyonlar şunlardır:

- **`printf()`:** Ekrana çıktı yazdırmak için kullanılır.

  Örnek:

  ```c
  #include <stdio.h>

  int main() {
    int sayi = 10;
    float ondalikSayi = 3.14;
    char karakter = 'A';
    char metin[] = "Merhaba";

    printf("Sayı: %d\n", sayi);
    printf("Ondalık Sayı: %.2f\n", ondalikSayi);
    printf("Karakter: %c\n", karakter);
    printf("Metin: %s\n", metin);

    return 0;
  }
  ```

- **`scanf()`:** Kullanıcıdan girdi almak için kullanılır.

  Örnek:

```c
#include <stdio.h>

int main() {
  int sayi;

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

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

  return 0;
}
```

- **`getchar()`:** Standart girişten (genellikle klavye) tek bir karakter okur ve bu karakteri `int` türünde bir değer olarak döndürür.

  Örnek:

```c
#include <stdio.h>

int main() {
  char karakter;

  printf("Bir karakter girin: ");
  karakter = getchar();

  printf("Girdiğiniz karakter: %c\n", karakter);

  return 0;
}
```

- **`putchar()`:** Belirtilen karakteri standart çıktıya (genellikle ekran) yazar. Başarı durumunda yazdırılan karakteri, hata durumunda ise EOF (End-of-File) değerini döndürür.

  Örnek:

```c
#include <stdio.h>

int main() {
  char karakter = 'A';

  putchar(karakter); // Ekrana 'A' karakterini yazar
```

```
  return 0;
}
```

4.2.2 <math.h>: Matematiksel İşlemler

<math.h> başlık dosyası, matematiksel işlemler için fonksiyonlar sağlar. En sık kullanılan fonksiyonlar şunlardır:

- **sqrt(x):** x sayısının karekökünü hesaplar. double türünde bir değer döndürür.

  Örnek:

```
#include <stdio.h>
#include <math.h>

int main() {
  double sayi = 25.0;
  double karekok = sqrt(sayi);

  printf("%.2f sayısının karekökü: %.2f\n", sayi, karekok);

  return 0;
}
```

- **pow(x, y):** x sayısının y'inci kuvvetini hesaplar. double türünde bir değer döndürür.

  Örnek:

```
#include <stdio.h>
#include <math.h>

int main() {
  double taban = 2.0;
  double us = 3.0;
  double sonuc = pow(taban, us);

  printf("%.2f sayısının %.2f. kuvveti: %.2f\n", taban, us, sonuc);

  return 0;
}
```

- **sin(x), cos(x), tan(x):** x açısının (radyan cinsinden) sinüs, kosinüs ve tanjant değerlerini hesaplar. double türünde bir değer döndürür.

  Örnek:

```c
#include <stdio.h>
#include <math.h>

int main() {
  double aci = 30.0; // Derece cinsinden
  double radyan = aci * M_PI / 180.0; // Dereceyi radyana çevirme

  printf("sin(%.2f): %.2f\n", aci, sin(radyan));
  printf("cos(%.2f): %.2f\n", aci, cos(radyan));
  printf("tan(%.2f): %.2f\n", aci, tan(radyan));

  return 0;
}
```

- **`double` Veri Tipi:** `double` veri tipi, `float` gibi ondalıklı sayıları temsil eder, ancak `float`'a göre **daha fazla hassasiyet** sağlar. `double`, genellikle 8 bayt bellek kullanırken, `float` 4 bayt kullanır.
- **`%.2lf` Format Belirteci:** `%.2lf`, `printf` fonksiyonunda `double` tipindeki bir değişkeni yazdırırken, virgülden sonra **iki basamak** göstermek için kullanılır. `lf`, "long float" anlamına gelir ve `double` veri tipi için kullanılır.

4.2.3 `<stdlib.h>`: Genel Amaçlı Fonksiyonlar

`<stdlib.h>` başlık dosyası, bellek yönetimi, sayı dönüştürme, rastgele sayı üretme gibi genel amaçlı fonksiyonlar sağlar. En sık kullanılan fonksiyonlar şunlardır:

- **`rand()`:** Rastgele bir sayı üretir.

  Örnek:

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  int rastgeleSayi = rand();
  printf("Rastgele sayı: %d\n", rastgeleSayi);
  return 0;
}
```

- **`srand(seed)`:** Rastgele sayı üreteci için başlangıç değeri (seed) ayarlar.

  Örnek:

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

int main() {
  srand(time(NULL)); // Her çalıştırmada farklı rastgele sayılar üretir

  for (int i = 0; i < 5; i++) {
    int rastgeleSayi = rand() % 100; // 0 ile 99 arasında rastgele sayı
    printf("%d ", rastgeleSayi);
  }
  printf("\n");
  return 0;
}
```

- **`time(NULL)` Fonksiyonu:** `time(NULL)` ifadesinde, `time()` fonksiyonuna `NULL` değeri parametre olarak verilir. Bu, fonksiyonun zaman bilgisini sadece dönüş değeri olarak döndürmesini ve ek bir yere yazmamasını sağlar.
- **`NULL` Değeri:** `NULL`, C dilinde bir işaretçinin hiçbir yeri göstermediğini belirtmek için kullanılan özel bir değerdir.
- **`abs(x)`:** x tam sayısının mutlak değerini hesaplar ve sonucu `int` türünde döndürür.

  Örnek:

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  int sayi = -15;
  int mutlakDeger = abs(sayi);
  printf("%d sayısının mutlak değeri: %d\n", sayi, mutlakDeger);
  return 0;
}
```

- **`atoi(str)`:** `str` stringini tam sayıya (`int`) dönüştürür.

  Örnek:

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  char metin[] = "123";
  int sayi = atoi(metin);
```

```c
  printf("String: %s, Tam Sayı: %d\n", metin, sayi);
  return 0;
}
```

4.2.4 `<time.h>`: Zaman ve Tarih İşlemleri

`<time.h>` başlık dosyası, zaman ve tarih işlemleri için fonksiyonlar sağlar. En sık kullanılan fonksiyonlar şunlardır:

- **`time()`:** Geçerli zamanı saniye cinsinden döndürür (1 Ocak 1970'den beri geçen saniye sayısı).

  Örnek:

```c
#include <stdio.h>
#include <time.h>

int main() {
  time_t suAn = time(NULL);
  printf("Şu anki zaman (saniye cinsinden): %ld\n", suAn);
  return 0;
}
```

- **Unix Epoch:** 1 Ocak 1970, Unix işletim sisteminin ve birçok programlama dilinin zamanı hesaplamak için kullandığı başlangıç noktasıdır. Bu tarihe **"Unix epoch"** denir.
- **`localtime()`:** `time_t` türündeki bir zaman bilgisini, yerel zaman dilimine göre yapılandırılmış bir `struct tm` yapısına dönüştürür.

  Örnek:

```c
#include <stdio.h>
#include <time.h>

int main() {
  time_t suAn = time(NULL);
  struct tm *zamanBilgisi = localtime(&suAn);

  printf("Yıl: %d\n", zamanBilgisi->tm_year + 1900);
  printf("Ay: %d\n", zamanBilgisi->tm_mon + 1);
  printf("Gün: %d\n", zamanBilgisi->tm_mday);

  return 0;
}
```

4.3 Rastgele Sayı Üretimi

- `<stdlib.h>` başlık dosyasında bulunan `rand()` fonksiyonu, sözde rastgele sayılar üretmek için kullanılır.
- `rand()` fonksiyonu, her çağrıldığında 0 ile `RAND_MAX` arasında bir tam sayı döndürür (`RAND_MAX` stdlib.h'de tanımlanmış bir makrodur).
- `srand(seed)` fonksiyonu, rastgele sayı üreteci için bir başlangıç değeri (seed) ayarlar.
- Farklı rastgele sayı dizileri elde etmek için `srand()` fonksiyonunu genellikle `time()` fonksiyonu ile birlikte kullanırız.

Örnek: Zar Atma Simülasyonu

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

int main() {
  srand(time(NULL)); // Rastgele sayı üreteci başlatılır
  int zar = (rand() % 6) + 1; // 1 ile 6 arasında rastgele bir sayı üretir
  printf("Zar: %d\n", zar);
  return 0;
}
```

4.4 Alıştırmalar

1. Kullanıcıdan bir sayı girmesini isteyin ve bu sayının karekökünü hesaplayıp ekrana yazdırın. (`sqrt()` fonksiyonunu kullanın).
2. Kullanıcıdan iki sayı girmesini isteyin (taban ve üs) ve bu sayıların üssünü hesaplayıp ekrana yazdırın. (`pow()` fonksiyonunu kullanın).
3. Rastgele bir sayı üretin (0 ile 100 arasında) ve bu sayının tek mi çift mi olduğunu ekrana yazdırın.
4. Kullanıcıdan bir saniye cinsinden zaman değeri girmesini isteyin (örneğin, 1 Ocak 1970'den beri geçen saniye sayısı) ve bu zamanı yerel zaman dilimine göre tarih ve saat formatında ekrana yazdırın. (`localtime()` fonksiyonunu kullanın).
5. Kullanıcıdan bir tam sayı girmesini isteyin ve bu sayının mutlak değerini hesaplayıp ekrana yazdırın. (`abs()` fonksiyonunu kullanın).
6. Zar atma simülasyonu yapan bir program yazın. Program her çalıştırıldığında 1 ile 6 arasında rastgele bir sayı üretsin.
7. 0 ile 99 arasında 10 tane rastgele sayı üreten bir program yazın.
8. Bir önceki soruda rastgele üretilen 10 sayının ortalamasını bulup ekrana yazdırın.

9. Kullanıcıdan 1 ile 12 arasında bir ay numarası girmesini isteyin ve o aya karşılık gelen kaç gün olduğunu ekrana yazdırın. (Şubat ayı için 28 gün olduğunu varsayın).
10. Kullanıcıdan Fahrenheit cinsinden sıcaklık değerini alıp Celsius'a çeviren bir program yazın. (Dönüşüm formülünü internette arayabilirsiniz.)
11. Kullanıcıdan bir kenar uzunluğu girmesini isteyin (ondalıklı sayı olabilir) ve bir küpün hacmini hesaplayıp ekrana yazdırın.
12. 0 ile 10 arasında rastgele 5 sayı üretin ve bu sayıların en büyüğünü ekrana yazdırın.
13. Kullanıcıdan doğum yılını girmesini isteyin ve yaşını hesaplayıp ekrana yazdırın.
14. Kullanıcıdan iki tarih bilgisi girmesini isteyin (saniye cinsinden, Unix zaman damgası formatında). Bu iki tarih arasındaki farkı gün olarak hesaplayan bir program yazın.
15. Rastgele 10 tam sayı üretin (0-100 arasında) ve bu sayılardan kaç tanesinin 50'den büyük olduğunu ekrana yazdırın.

4.5 Sonuç

C'nin standart kütüphanesi, programcıların hayatını kolaylaştıran birçok hazır fonksiyon sağlar. Bu fonksiyonları kullanarak, programcılar karmaşık işlemleri daha kolay ve verimli bir şekilde gerçekleştirebilirler. Bu bölümde, C'nin standart kütüphanesindeki bazı önemli fonksiyonları ve bunların nasıl kullanılacağına dair örnekler gördük. Bu fonksiyonları öğrenerek, daha güçlü ve etkili C programları yazabilirsiniz.