Nesne Yönelimli Dosya İşlemleri

Bilgisayar Programlama Dersi

Prof.Dr.Kasım KURT

Fizik Bölümü

10 Aralık 2025

İçerik

Fizikte Neden Dosya İşlemleri?

Fiziksel simülasyonlar ve deneyler genellikle anlık ekrana basılamayacak kadar çok veri üretir.

- **Veri Saklama:** Bir simülasyonun (örn. gezegen yörüngesi) sonuçlarını daha sonra analiz etmek için (‘.txt‘, ‘.csv‘, ‘.dat‘).
- **Konfigürasyon:** Simülasyon parametrelerini (başlangıç hızı, kütle) kodun içine gömmek yerine dosyadan okumak.
- **Otomasyon:** Binlerce deney sonucunu otomatik olarak işlemek.

Eski Yöntem vs. Nesne Yönelimli Yaklaşım

Klasik (C-Tarzı / Prosedürel)

- Dosya aç/kapa işlemleri ‘main‘ içinde yapılır.
- ‘dosya.close()‘ unutulursa veri kaybı olabilir.
- Kod karmaşıktır ve tekrar kullanımı zordur.

Nesne Yönelimli (OOP)

- Dosya işlemleri bir Sınıf (Class) içine gizlenir.
- **Constructor:** Nesne doğduğunda dosya açılır.
- **Destructor:** Nesne öldüğünde dosya otomatik kapanır (RAII).
- Daha güvenli ve temiz kod.

Temel Kütüphane: <fstream>

C++'ta dosya işlemleri için üç temel sınıf kullanılır:

- ofstream (Output File Stream): Dosyaya veri **yazmak** için.
- ifstream (Input File Stream): Dosyadan veri **okumak** için.
- fstream : Hem okuma hem yazma için.

Örnek Tanımlama:

```
1       #include <fstream>
2       ...
3       std::ofstream dosya("deney_sonuc.txt");
4       dosya << "Zaman t=0" << std::endl;
5
```

Sınıf Tasarımı: VeriKaydedici

Fiziksel verileri kaydeden sınıfımızın yapısı:

```
1       class DeneyVeriKaydedici {
2         private:
3         ofstream dosyaAkisi; // Ak     nesnesi gizli (
    Private)
4
5         public:
6         // Kurucu: Dosyay  a ar
7         DeneyVeriKaydedici(string isim) {
8           dosyaAkisi.open(isim);
9           // Hata kontrol  burada yap l r
10          if(dosyaAkisi.is_open()) {
11            dosyaAkisi << "# t(s)  x(m)  v(m/s)" << endl;
12          }
13        }
14        // ...
15
```

Otomatik Kaynak Yönetimi (Destructor)

En önemli kısım: Dosyanın kapatılmasının garanti altına alınması.

```
1        // Y k c Fonksiyon (Destructor)
2        // Nesne kapsamdan ktnda otomatik
         alr .
3        ~DeneyVeriKaydedici() {
4          if (dosyaAkisi.is_open()) {
5            dosyaAkisi.close();
6            cout << "Dosya guvenle kapatildi." << endl;
7          }
8        }
9
10       // Veri Yazma Metodu
11       void veriEkle(double t, double x, double v) {
12         dosyaAkisi << t << "\t" << x << "\t" << v << endl;
13       }
14     };
15
```

Kullanım (Main Fonksiyonu)

Fiziksel simülasyon döngüsü içinde kullanımı oldukça basittir:

```
int main() {
  // 1. Nesneyi olu tur (Dosya a   ld  )
  DeneyVeriKaydedici kaydedici("atiss_deneyi.txt");

  double t = 0.0, y = 100.0, g = 9.81;

  // 2. Sim lasyon D  ng  s
  while(y > 0) {
    y = 100.0 - 0.5 * g * t * t;

    // Veriyi nesneye g nder , gerisini o halletsin
    kaydedici.veriEkle(t, y, g*t);

    t += 0.1;
  }

  return 0;
  // Program bitti inde 'kaydedici' yok edilir
  // ve dosya otomatik kapan r.
}
```

Özet

Bu derste şunları öğrendik:

- `ofstream` kullanarak veri yazmayı.
- Dosya işlemlerini bir sınıf (Class) içine alarak kod karmaşasını önlemeyi.
- **RAII Prensibi:** Dosya açma işlemini `Constructor`'da, kapatma işlemini `Destructor`'da yaparak hata riskini sıfıra indirmeyi.

Ödev: Bu sınıfı, dosyadan veri OKUYACAK (ifstream) şekilde 'VeriOkuyucu' adıyla yeniden yazınız.