

# C++'ta Veri Kalıcılığı

## Dosya Akışları ile Sağlam Yapılar İnşa Etmek

Prof.Dr. Kasım KURT

1 Aralık 2025

# Projenin Amacı: Neden “İnşa Etmeliyiz”?

## GEÇİCİ (RAM)

Programlarımız neden her kapandığında her şeyi unutuyor?

Veriler ana bellekte (RAM) hapsolmuş durumda ve program bittiğinde kaybolur.

## KALICI (Disk)

**Çözüm: Veri Dosyaları.**

Verileri program dışında, disk gibi kalıcı bir ortamda saklamanın yolu.

## Metin Dosyaları (.txt, .cpp)

- İnsan tarafından okunabilir.
- Not defteri gibi programlarla açılabilir.
- *Bu dersteki odak noktamız.*

## Binary Dosyalar (.dat, .exe)

- Makine tarafından okunabilir (010101...).
- Genellikle daha kompakt ve hızlıdır.
- Sayılar gerçek ikili formlarında saklanır.

## Akış (Stream) Kavramı

Bir program ile dosya arasındaki tek yönlü veri iletim yolu.

### Temel Araçlarımız:

- `ifstream`: Dosyadan programa veri okumak için (**I**nput **F**ile **S**tream).
- `ofstream`: Programdan dosyaya veri yazmak için (**O**utput **F**ile **S**tream).

Bu araçları kullanmak için kütüphane: `#include <fstream>`

# Adım 1: Kapatı Açmak ve Güvenliğı Sağlamak

```
1 // 1. Bir cikis akis nesnesi olustur
2 ofstream outFile;
3
4 // 2. Nesneyi harici bir dosyaya bagla
5 outFile.open("prices.dat");
6
7 // 3. Olmazsa olmaz: Guvenlik kontrolu yap
8 if (outFile.fail()) {
9     cout << "Dosya acilamadi!";
10    exit(1); // Programi sonlandir
11 }
12
```

- Dosya açma işlemi başarısız olabilir (dosya bulunamaz, izinler yetersiz vb.).
- Her open() çağrısından sonra bu kontrolü yapmak zorunludur!
- fail() fonksiyonu, açma işlemi başarısız olursa true döner.

# İnteraktif Mola: Hata Avı!

Bu kod neden çökebilir veya beklenmedik davranışlar sergileyebilir?

```
1  #include <iostream>
2  #include <fstream>
3
4  int main() {
5      ifstream inFile;
6      inFile.open("var_olmayan_dosya.txt");
7
8      int value;
9      inFile >> value; // Hata burada olabilir!
10
11     cout << "Deger: " << value << endl;
12     return 0;
13 }
14
```

**Sorun:** Güvenlik kontrolü eksik! Dosya açılmazsa program hatalı çalışır.

**İşi Hızlandırmak:** Tanımlama ve açma işlemini tek adımda yapabilirsiniz.

```
1      ofstream outFile("prices.dat");  
2
```

**Dinamik Dosya Adları:** Dosya adını kodun içine gömmek yerine kullanıcıdan alın.

```
1      string filename;  
2      cout << "Dosya adını girin: ";  
3      cin >> filename;  
4  
5      // c_str() fonksiyonu string nesnesini C-tarzi karaktere donusturur.  
6      ifstream inFile(filename.c_str());  
7
```

# İşi Bitirmek: Dükkanı Kapatmak

İşiniz bittiğinde kaynakları serbest bırakmak için dosyayı `close()` ile kapatmalısınız.

```
1   outFile.close();  
2   inFile.close();  
3
```

- Bu, işletim sisteminin aynı anda açık tutabileceği dosya sayısı limiti için önemlidir.
- İyi bir programlama alışkanlığıdır.

# Veri Döşemek: Dosyaya Yazma

Tıpkı cout ile ekrana yazdığınız gibi! Tek fark, hedef nesnenin cout yerine dosya akış nesneniz olmasıdır.

```
1 // Ekrana yazar: cout << "Mats" << 39.95 << endl;  
2  
3 // Dosyaya yazar:  
4 outFile << "Mats" << 39.95 << endl;  
5 outFile << "Bulbs" << 3.22 << endl;  
6 outFile << "Fuses" << 1.08 << endl;  
7
```

## Dosya İçeriği (prices.dat)

Mats 39.95

Bulbs 3.22

Fuses 1.08

**Kelime Kelime Okuma (>>)** Tıpkı cin gibi çalışır. Boşluk, tab veya yeni satır karakterinde durur.

```
1      string descrip;
2      double price;
3
4      // En iyi EOF kontrol yontemi:
5      while (inFile >> descrip >>
6      price) {
7          cout << descrip << " " <<
8      price << endl;
9      }
```

**Tüm Satırı Okuma (getline)** Boşluk içeren metinleri veya tüm satırı tek seferde okumak için kullanılır.

```
1      string line;
2
3      while (getline(inFile, line))
4      {
5          cout << line << endl;
6      }
```

**Sıralı Erişim (Sequential Access):** Dosyayı baştan sona okumak.

**Rastgele Erişim (Random Access):** Doğrudan dosyanın ortasındaki bir bayta atlamak!

- `seekg()` & `seekp()`: Okuma (get) veya yazma (put) imlecini taşır.
- `tellg()` & `tellp()`: İmlecin mevcut konumunu döndürür.

```
1 // Bastan 10. bayta git
2 inFile.seekg(10L, ios::beg);
3
4 // Mevcut konumdan 5 bayt geri git
5 inFile.seekg(-5L, ios::cur);
6
7 // Dosyanin sonuna git
8 inFile.seekg(0L, ios::end);
9
```

# Modüler İnşaat: Akışları Fonksiyonlara Geçmek

Kodunuzu düzenli tutmak için dosya akış nesnelerini fonksiyonlara parametre olarak geçirebilirsiniz.

## Anahtar Kural

Her zaman **referans (&)** olarak geçilmelidir!

```
1 // Fonksiyon prototipi referans parametresi alır
2 void writeData(ofstream& file);
3
4 int main() {
5     ofstream myFile("data.txt");
6     if (!myFile.fail()) {
7         writeData(myFile); // Nesneyi fonksiyona gecir
8     }
9     myFile.close();
10 }
11
```

**Problem Senaryosu:** Son 10 polen sayımını tutan bir dosyamız var. Yeni bir sayım geldiğinde, en eski olanı silip yeni olanı sona ekleyerek dosyayı güncellememiz gerekiyor.

## Algoritma (Plan):

- 1 Giriş (pollen.in) ve çıkış (pollen.out) dosyalarını aç.
- 2 Kullanıcıdan yeni polen sayımını al.
- 3 Giriş dosyasından en eski sayımı oku (ama kullanma/atla).
- 4 Kalan 9 sayımı döngüyle oku ve doğrudan çıkış dosyasına yaz.
- 5 Yeni sayımı çıkış dosyasının sonuna yaz.
- 6 Dosyaları kapat ve sonucu bildir.

# Vaka Analizi: Kodun İnşası

```
1  double pollenUpdate (ifstream& inFile, ofstream& outFile) {
2      const int POLNUMS = 10;
3      int oldreading, polreading, newcount;
4      double sum = 0;
5      cout << "Yeni polen sayimini girin: ";
6      cin >> newcount;
7      // En eski veriyi oku ve atla
8      inFile >> oldreading;
9      // Kalan verileri oku, topla ve yeni dosyaya yaz
10     for (int i = 1; i < POLNUMS; i++) {
11         inFile >> polreading;
12         sum += polreading;
13         outFile << polreading << endl;
14     }
15     // Yeni veriyi sona ekle
16     outFile << newcount << endl;
17     sum += newcount;
18     inFile.close(); outFile.close();
19     return sum / POLNUMS;
```

# Proje Tamamlandı: Sonuç

pollen.in (Önce)

```
30 (Eski)
40
80
...
170
```



pollen.out (Sonra)

```
40
80
...
170
200 (Yeni)
```

En eski veri (30) başarıyla kaldırıldı ve en yeni veri (200) eklendi.

# Kaçınılması Gereken İnşaat Kazaları

- X Hata:** Dosyayı açmadan okuma/yazma yapmak.
- ✓ **Kural:** Her zaman önce `open()` çağırın.
- X Hata:** `open()` sonrası `fail()` kontrolünü unutmak.
- ✓ **Kural:** Başarılı bir bağlantıdan emin olun.
- X Hata:** Boşluk içeren verileri `»` ile okumaya çalışmak.
- ✓ **Kural:** Tüm satır için `getline` kullanın.
- X Hata:** Çıkış dosyasının (`ofstream`) üzerine yazılacağını unutmak.
- ✓ **Kural:** Silmek istemiyorsanız `ios::app` (ekleme) modunu kullanın.

# Bugün Neler İnşa Ettik?

- ❶ **Temelleri Attık:** Veri dosyaları ile kalıcı depolamanın 'neden' ve 'nasıl' olduğunu anladık.
- ❷ **Araçları Kullandık:** `ifstream` ve `ofstream` ile veri akışını yönettik.
- ❸ **Yapıyı Kurduk:** Dosyaları güvenli bir şekilde açtık, okuduk, yazdık ve kapattık.
- ❹ **Gelişmiş Özellikler:** `seekg` ile gezindik ve fonksiyonlara referans ile geçtik.
- ❺ **Proje:** Polen sayımı vaka analizi ile öğrendiklerimizi birleştirdik.

*Artık verileriniz için kalıcı ve sağlam yapılar inşa etmeye hazırsınız.*