C++ İle Dosya İşlemleri ve Yapısal Programlama

Öğrenci Kayıt Sistemi Örneği Üzerinden İnceleme

PRof.Dr.Kasım KURT

Fen Fakültesi Fizik Bölümü

8 Aralık 2025

Sunum İçeriği

Proje Hakkında

Amaç:

- C++ kullanarak kalıcı bir veri saklama sistemi oluşturmak.
- Öğrenci bilgilerini (No, Ad, Soyad, Bölüm) yönetmek.

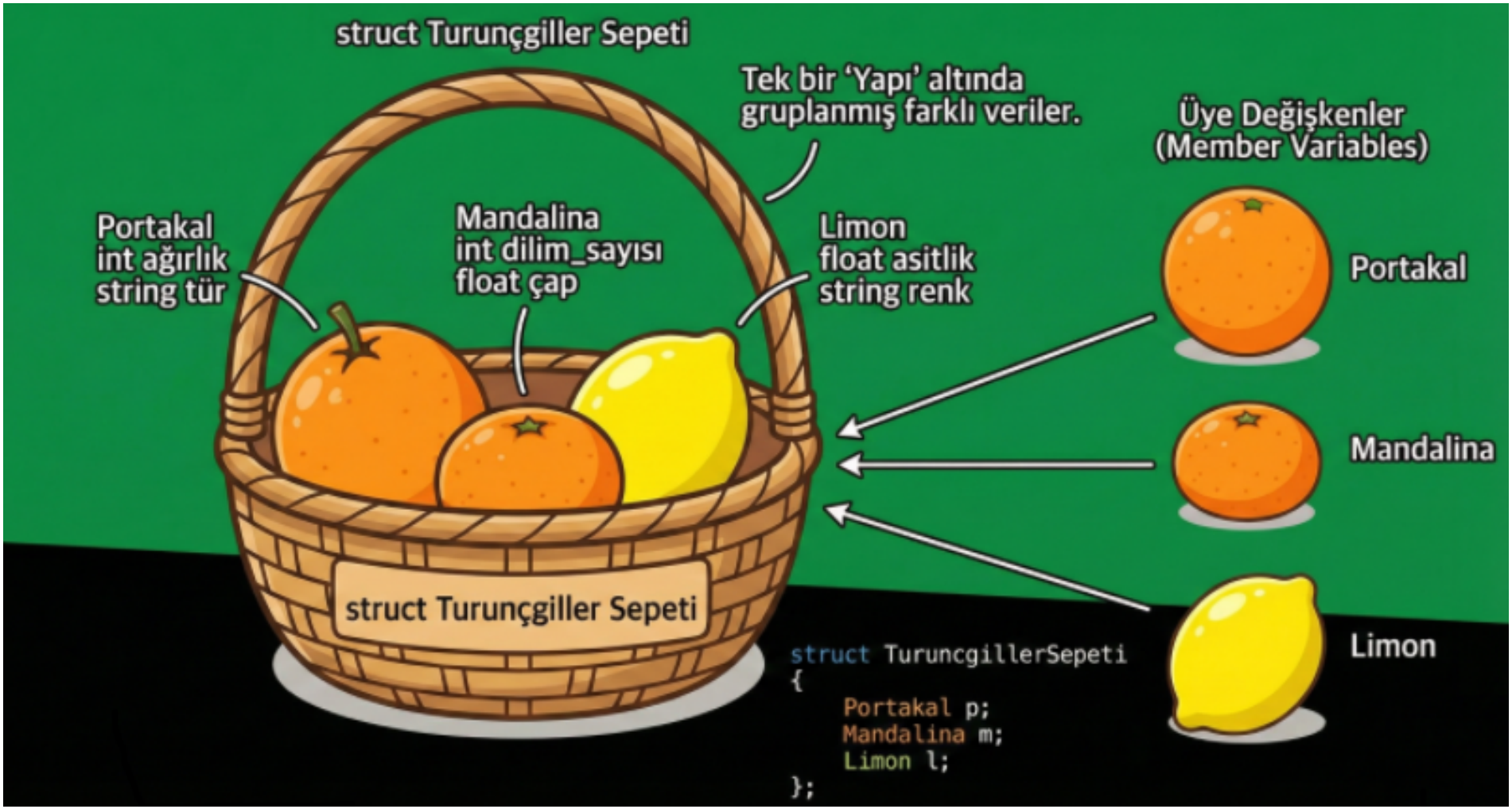
Kullanılan Temel Teknolojiler:

- `struct` (Yapılar)
- `fstream` (Dosya Giriş/Çıkış)
- Global ve Lokal Değişken Yönetimi

Yapı (Struct) Nedir?

- C++ dilinde, farklı veri tiplerini (int, string, float vb.) tek bir çatı altında toplamak için kullanılan kullanıcı tanımlı veri tipidir.
- Bir öğrencinin numarası (int) ve adı (string) birbirinden farklı tiplerdir ama mantıksal olarak bir bütündür.
- `struct`, bu verileri paketlememizi sağlar.

Yapı (struct) örnek



Kod İçindeki Struct Yapısı

Projemizde kullandığımız yapı şöyledir:

```
1       struct Ogrenci {
2         int numara;        // Tamsayi verisi
3         string ad;         // Metin verisi
4         string soyad;      // Metin verisi
5         string bolum;      // Metin verisi
6       };
7
```

Bu tanım sayesinde `Ogrenci` adında yeni bir veri tipi oluşturmuş olduk.

Struct Kullanımı ve Erişim

Bir struct tanımlandıktan sonra, ondan bir nesne (değişken) türetilir.

```
1       // Nesne Turetiyoruz
2       Ogrenci ogr;
3
4       // Veriye Erisim (Nokta operatoru ile)
5       ogr.numara = 101;
6       ogr.ad = "Ali";
7
```

Neden Önemli?

Eğer struct kullanmasaydık, her öğrenci özelliği için ayrı diziler (array) tutmak zorunda kalırdık. Struct kodun okunabilirliğini artırır.

Değişken Kapsamı (Scope) Nedir?

Bir değişkenin programın neresinden erişilebilir olduğunu belirleyen kurallardır. İki temel türü inceleyeceğiz:

1. Global Değişkenler
2. Lokal (Yerel) Değişkenler

Global Değişkenler

Tanım: Fonksiyonların dışında, genellikle dosyanın en başında tanımlanan değişkenlerdir.
Özellikleri:

- Programın her yerinden (tüm fonksiyonlardan) erişilebilir.
- Program çalıştığı sürece bellekte yer kaplar.

Projemizdeki Global Değişken

Projemizde dosya adını global olarak tanımladık:

```
1       const string DOSYA_ADI = "ogrenciler.txt";
2
3       void ogrenciEkle() { ... }
4       void ogrenciSil() { ... }
5
```

Neden Global Yaptık?

- `Ekle`, `Listele` ve `Sil` fonksiyonlarının hepsi aynı dosyaya işlem yapıyor.
- Dosya adını değiştirmek istersek, tek bir satırı değiştirmemiz yeterli olacaktır.

Lokal (Yerel) Değişkenler

Tanım: Bir fonksiyonun veya bir bloğun { } içinde tanımlanan değişkenlerdir.
Özellikleri:

- Sadece tanımlandığı fonksiyon içinden erişilebilir.
- Fonksiyon bittiğinde bellekten silinirler.
- **Güvenlik:** Diğer fonksiyonlar bu değişkeni yanlışlıkla değiştiremez.

Projemizdeki Lokal Değişkenler

```
1       void ogrenciEkle() {
2         Ogrenci ogr; // Sadece bu fonksiyonda gecerli
3         // ...
4       }
5
6       int main() {
7         int secim;    // Sadece main icinde gecerli
8         // ...
9       }
10
```

Not: Main içindeki ‘secim‘ değişkenini ‘ogrenciEkle‘ fonksiyonu göremez.

Dosya İşlemleri Kütüphanesi: fstream

C++'ta dosya işlemleri için `<fstream>` kütüphanesi kullanılır. Üç ana sınıf vardır:

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

Dosya Açma Modları

Dosyayı hangi amaçla açtığımızı belirtmemiz gerekir:

- `ios::in` → Okuma modu (Varsayılan ifstream modu).
- `ios::out` → Yazma modu (Dosya içeriğini siler ve baştan yazar).
- `ios::app` → **Append (Ekleme)** modu. Dosyanın sonuna ekler, eski verileri korur.

Dosyaya Yazma (Ekleme İşlemi)

Veri kaybını önlemek için `ios::app` kullanıyoruz.

```
1       // Dosya yazma modunda acilir (Append)
2       ofstream dosya(DOSYA_ADI.c_str(), ios::app);
3
4       if (dosya.is_open()) {
5         dosya << ogr.numara << " " << ogr.ad << "\n";
6         dosya.close();
7       }
8
```

Dikkat

Dosya ile işimiz bittiğinde `.close()` ile kapatmak zorundayız. Aksi takdirde veriler kaydedilmeyebilir.

Dosyadan Okuma (Listeleme İşlemi)

Dosyayı baştan sona okumak için `while` döngüsü kullanılır.

```
1       ifstream dosya(DOSYA_ADI.c_str());
2
3       // >> operatoru bosluk gorene kadar okur
4       while (dosya >> numara >> ad >> soyad >> bolum) {
5         cout << numara << "\t" << ad << ...;
6       }
7
```

Bu döngü, dosya sonuna (End Of File - EOF) gelene kadar otomatik olarak döner.

Dosyadan Veri Silme Mantığı

Dosyaların ortasından doğrudan bir satır silmek mümkün değildir. Bu yüzden **"Kopyala - Atla - Değiştir"** yöntemi kullanılır.

Algoritma Adımları:

1. Orijinal dosyayı **Oku** modunda aç.
2. `gecici.txt` adında yeni bir dosya oluştur.
3. Orijinal dosyadaki verileri tek tek oku.
4. Silinecek veri **hariç** diğerlerini geçici dosyaya yaz.
5. Silinecek veriyi yazma (atla).

Dosya Silme ve Yeniden Adlandırma

Kopyalama işlemi bittikten sonra eski dosya silinmeli ve geçici dosyanın adı düzeltilmelidir.

```
1       okuDosya.close();
2       yazDosya.close();
3
4       // <cstdio> kutuphanesinden gelirler
5       remove(DOSYA_ADI.c_str());          // Eskiyi sil
6       rename("gecici.txt", DOSYA_ADI.c_str()); // Yeninin adini
      degistir
7
```

Menü Tasarımı

Kullanıcının programda sürekli kalabilmesi için sonsuz döngü (`while(true)`) kullanılır.

```
1       while (true) {
2         cout << "1. Ekle  2. Sil ...";
3         cin >> secim;
4
5         switch (secim) {
6           case 1: ekle(); break;
7           // ...
8           case 4: return 0; // Programdan cikis
9         }
10      }
11
```

Hata Kontrolü (Error Handling)

Dosya işlemlerinde her zaman dosyanın açılıp açılmadığını kontrol etmeliyiz.

- Dosya başka bir program tarafından kullanılıyor olabilir.
- Disk dolu olabilir veya yazma izni olmayabilir.

Bu yüzden kodumuzda her zaman şu kontrolü yaptık: `if (dosya.is_open()) { ... }`

Sonuç ve Özet

Bu projede şunları öğrendik:

- **Struct:** Verileri düzenli tutmak için.
- **Global/Local Değişkenler:** Veri güvenliği ve erişim yönetimi için.
- **Ofstream/Ifstream:** Kalıcı veri saklamak için.
- **Algoritma:** Dosyadan veri silme mantığını kurmak için.

Geliştirme Önerileri

Bu projeyi daha da geliştirmek için neler yapılabilir?

1. Öğrenci ismine veya numarasına göre **Arama** fonksiyonu eklenebilir.
2. Öğrenci bilgilerini **Güncelleme** (silip tekrar ekleme mantığıyla) yapılabilir.
3. Not ortalaması gibi hesaplamalar eklenebilir.

Teşekkürler!

Sorularınız?