# Nesne Yönelimli Programlama Ödevi Raporu

434389 Zehra Nur Bayav
434407 Sedanur Şeker

Bu çalışmada C++ programlama dili kullanılarak dinamik dizi yapısını temel alan bir sınıf tasarımı gerçekleştirilmiştir. Çalışmanın amacı; nesne yönelimli kavramlarını uygulamalı olarak öğrenmek ve dinamik bellek yönetimi, miras ve operatör aşırı yükleme gibi temel konuları pekiştirmektir.
Proje kapsamında üç tane sınıf geliştirilmiştir. Bunlar:

- `DynamicArray`
- `SortedArray`
- `UniqueArray`

`DynamicArray` temel sınıf olarak tasarlanmış, `SortedArray` ve `UniqueArray` sınıfları bu sınıftan türetilerek özel davranışlar kazandırılmıştır. Bu yaklaşımla kod tekrarının önüne geçilmiş ve nesne yönelimli tasarım prensipleri etkin bir şekilde kullanılmıştır.

## DynamicArray Sınıfı

`DynamicArray` sınıfı, C++'ta sabit boyutlu dizilerin yetersiz kaldığı durumlar için dinamik olarak büyüyebilen bir dizi yapısı sunmaktadır. Bu sınıf, eleman sayısı arttıkça kapasitesini otomatik olarak artırabilen bir yapıdadır.
Sınıf içerisinde aşağıdaki private veri üyeleri tanımlanmıştır.

- `int* m_data`
  Dinamik olarak ayrılan bellek alanını tutan işaretçidir.

- `int m_size`
  Dizide bulunan mevcut eleman sayısını ifade eder.

- `int m_capacity`
  Ayrılmış olan toplam kapasiteyi belirtir.

## Contructors and Destructors:

**Default Constructor:** Dizi için başlangıç kapasitesi ayırır ve diziyi boş olarak oluşturur.
**Parametreli Constructor:** Kullanıcının belirttiği kapasiteye göre dinamik bellek ayırır.
**Copy Constructor:** Deep copy mantığı ile çalışarak yeni nesne için ayrı bir bellek alanı oluşturur.
**Destructor:** Dinamik olarak ayrılan belleği serbest bırakarak bellek sızıntılarını önler.

## DynamicArray İçindekiTemel Fonksiyonlar

- `push(int value)` : Diziye yeni eleman ekler.
- `pop()` : Dizinin son elemanını siler.
- `get(int index)` : Belirtilen indisteki elemanı döndürür.
- `set(int index, int value)` : Belirtilen indisteki değeri günceller.
- `clear()` : Diziyi tamamen temizler.
- `isEmpty()` : Dizinin boş olup olmadığını kontrol eder.
- `getSize()` : Mevcut eleman sayısını döndürür.
- `getCapacity()` : Dizinin kapasitesini döndürür.
- `print()` : Dizinin içeriğini ekrana yazdırır.

### Operatör Aşırı Yükleme

Projede C++'ın sunduğu operatör aşırı yükleme özelliği kullanılmıştır. Bu kapsamda:

- `operator[]` ile dizinin elemanlarına indeks kullanılarak erişim sağlanmıştır.
- `operator+` ile iki dizinin birleştirilmesi mümkün hale getirilmiştir.
- `operator=` atama işlemini deep copy mantığıyla gerçekleştirmektedir.
- `operator==` ve `operator!=` dizilerin içeriklerinin karşılaştırılmasını sağlamaktadır.

## SortedArray SINIFI

`SortedArray` sınıfı, `DynamicArray` sınıfından türetilmiş olup dizideki elemanların her zaman sıralı tutulmasını amaçlamaktadır.

```
class SortedArray : public DynamicArray
```

Bu yapı, bir inheritance uygulaması örneğidir.

### push fonksiyonunun Override Edilmesi

Bu sınıfta `push` fonksiyonu override edilmiştir. Yeni eklenen eleman, uygun konuma yerleştirilerek dizinin sıralı kalması sağlanmaktadır. Böylece kullanıcı tarafından eklenen elemanların sırası otomatik olarak düzenlenmektedir.

### Binary Search

`SortedArray` sınıfında ayrıca `binarySearch(int value)` fonksiyonu bulunmaktadır. Dizi sıralı olduğu için ikili arama algoritması kullanılmış ve arama işlemleri daha verimli hale getirilmiştir.

## UniqueArray Sınıfı

`UniqueArray` sınıfı, tekrar eden elemanlara izin vermeyen bir dizi yapısı sunmaktadır. Bu sınıf da `DynamicArray` sınıfından türetilmiştir.

contains Fonksiyonu**:**

`contains(int value)` fonksiyonu, verilen bir değerin dizide bulunup bulunmadığını kontrol eder.

`Push` fonksiyonu override ederek eleman dizide varsa ekleme yapılmaz ama eleman yoksa diziye eklenir.

Bu sayede dizideki tüm elemanların benzersiz olması garanti altına alınmıştır.

`main.cpp` dosyasında tüm sınıflar için nesneler oluşturulmuş ve yazılan fonksiyonlar test edilmiştir. Bu testler sayesinde sınıfların doğru ve beklenen şekilde çalıştığı gözlemlenmiştir.

## SONUÇ

Bu çalışmada C++ programlama dili kullanılarak dinamik dizi yapısı temel alınmış ve nesne yönelimli programlama prensipleri başarıyla uygulanmıştır. `DynamicArray` sınıfı temel bir yapı sunarken, `SortedArray` ve `UniqueArray` sınıfları bu yapıyı genişleterek farklı kullanım senaryoları oluşturmuştur.

Çalışma, dinamik bellek yönetimi, miras, polimorfizm ve operatör aşırı yükleme konularının anlaşılmasına katkı sağlamıştır ve C++ programlama becerilerinin geliştirilmesi açısından faydalı olmuştur.