Nesne Yönelimli Programlama

Nesne Yönelimli Programlama

Ders İçeriği

Hafta 2: Kontrol Yapıları ve Döngüler

- Koşul İfadeleri (`if` , `else if` , `else`)
- `switch-case` Yapısı
- Döngüler: `for` , `while` , `do-while`
- Break, Continue ve Return Kullanımı
- Nested (İç İçe) Döngüler

Giriş

Kontrol yapıları ve döngüler, programların kararlar almasını ve tekrarlayan işlemleri gerçekleştirmesini sağlar. Niçin kontrol yapıları ve döngüler kullanılır. Programlamada niçin önemlidir?

1. **Karar Verme ve Program Akışını Yönetme**

- **Kontrol yapıları** (if-else, switch-case) programların farklı durumlarda nasıl davranması gerektiğini belirler. Bu sayede programlar, kullanıcı girdilerine, değişkenlerin değerlerine veya farklı koşullara göre esnek ve dinamik tepkiler verebilir.
- Örneğin, bir not sisteminde kullanıcıdan alınan notun hangi harf notuna denk geldiği kontrol yapılarına göre belirlenir.

2. **Kodun Tekrarını Engelleme ve Verimlilik**

- **Döngüler** tekrarlı işlemlerin pratik bir şekilde yapılmasını sağlar. Aynı işlemi tekrar tekrar manuel olarak yazmak yerine, döngüleri kullanarak bu işlemleri otomatik hale getirebiliriz.
- Örneğin, bir dizinin tüm elemanlarını sıralamak, hesaplamak veya ekrana yazdırmak için döngüler kullanılır. Böylece aynı işlemi yüzlerce kez kodlamak yerine bir döngüyle tamamlamak mümkün olur.

3. **Programların Modüler ve Okunabilir Olmasını Sağlama**

- Kontrol yapıları ve döngüler, kodun düzenli ve modüler olmasına katkıda bulunur. Karmaşık problemler küçük parçalara ayrılarak kontrol yapılarına ve döngülere yerleştirilebilir.
- Bu da hem kodun okunabilirliğini artırır hem de hata ayıklamayı ve geliştirmeyi kolaylaştırır.

4. **Kullanıcı Girdilerine Dinamik Yanıt Verme**

- Programlar, kullanıcı girdisine göre farklı işlemler gerçekleştirmek zorundadır. Kontrol yapıları sayesinde kullanıcıdan gelen veriler analiz edilerek farklı yollar izlenir.
- Döngüler ise kullanıcıya sürekli aynı işlemi tekrar yaptırmak zorunda kalmadan dinamik girdilerle işlemlerin devam etmesini sağlar.

5. **Veri İşleme ve Büyük Miktarda Bilgi ile Çalışma**

- Döngüler büyük veri kümeleriyle çalışırken oldukça verimlidir. Bir dosyadaki verileri okuma, bir dizideki öğeleri sıralama, bir veritabanında kayıtları listeleme gibi işlemler döngülerle yapılır.
- Kontrol yapıları ise bu veri üzerinde işlem yapılırken farklı koşullara göre karar alınmasını sağlar.

6. **Otomasyon ve Tekrar Eden Görevler**

- Döngüler, bir işlemin belirli bir sayıda tekrarlanması gereken durumlarda kullanılır. Bu da programları otomatik hale getirir.
- Örneğin, bir hesaplama işlemini yüzlerce kez yapmanız gerektiğinde, döngüler sayesinde bu işlemi tek satırla çözmek mümkün olur.

7. **Programları Daha Akıllı ve Etkileşimli Yapma**

- Kontrol yapıları sayesinde programlar, koşullara göre farklı sonuçlar üreterek daha "akıllı" hale gelir.
- Örneğin, bir oyun programında oyuncunun kazandığı puana göre seviye atlama veya oyunun sonlanması kontrol yapılarıyla sağlanır.

8. **Daha Az Kodla Daha Fazla İş Yapabilme**

- Döngüler ve kontrol yapıları, daha az kod yazarak daha karmaşık işlemleri gerçekleştirmemizi sağlar. Aynı işlemi birden fazla yazmak yerine, döngüler sayesinde işlemleri verimli şekilde tekrarlarız.

Koşullu İfadeler

if-else Yapısı

`if-else` yapısı, bir koşulun doğruluğunu kontrol ederek programın iki farklı yolda ilerlemesini sağlar.

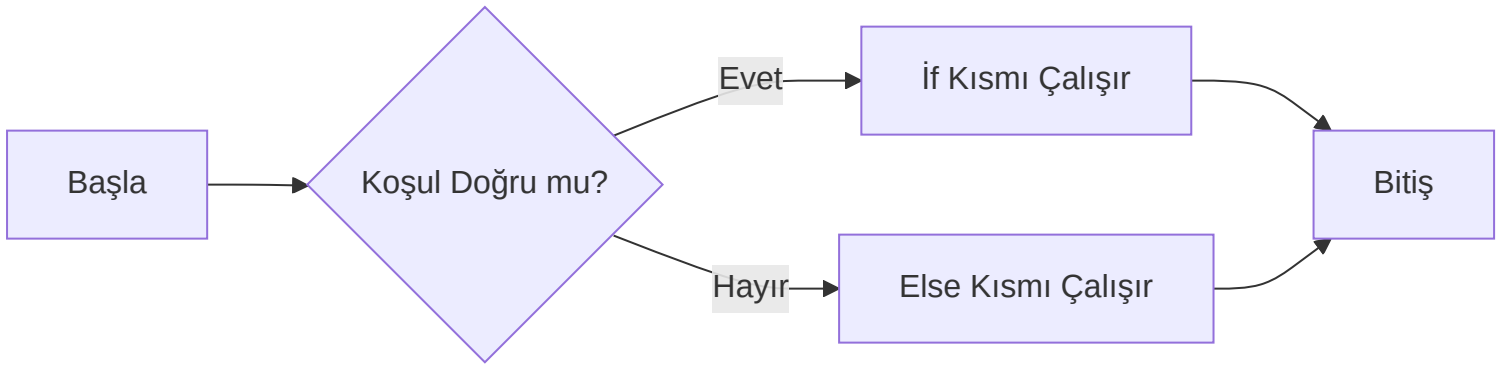
Yapı:

```
if (koşul) {
    // Koşul doğruysa yapılacak işlemler
} else {
    // Koşul yanlışsa yapılacak işlemler
}
```

```cpp
int sayi = 10;
if (sayi > 0) {
    cout << "Sayı pozitiftir." << endl;
} else {
    cout << "Sayı negatiftir veya sıfırdır." << endl;
}
```

else-if Yapısı

`else-if` yapısı, birden fazla koşulun kontrol edilmesi gerektiğinde kullanılır.



Kod Yapısı (Syntax):

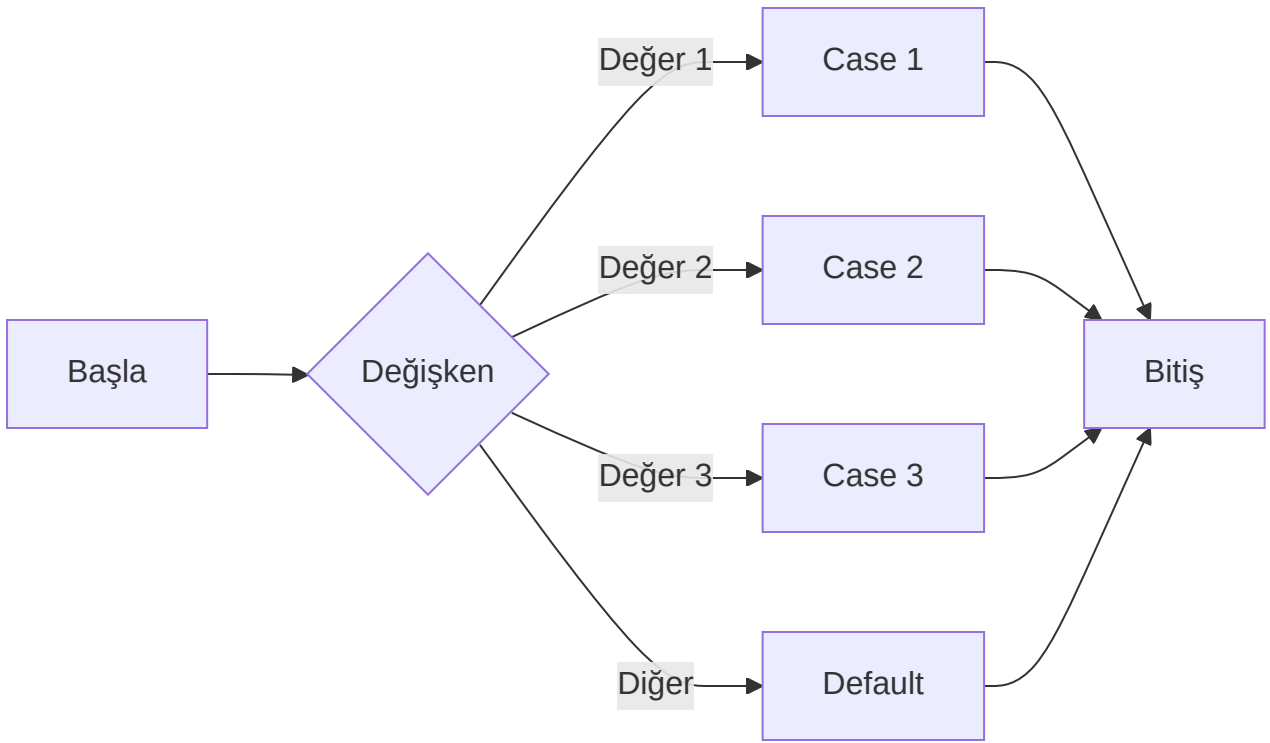
```
Kodu kopyala
if (koşul1) {
    // Koşul1 doğruysa yapılacak işlemler
} else if (koşul2) {
    // Koşul2 doğruysa yapılacak işlemler
} else {
    // Hiçbir koşul doğru değilse yapılacak işlemler
}
```

Örnek:

```
int notDegeri = 85;
if (notDegeri >= 90) {
    cout << "Harf Notu: A" << endl;
} else if (notDegeri >= 80) {
    cout << "Harf Notu: B" << endl;
} else {
    cout << "Harf Notu: C" << endl;
}
```

switch-case Yapısı

`switch-case` yapısı, bir değişkenin belirli bir değere sahip olup olmadığını kontrol eder ve bu değerlere göre farklı işlemler yapar.



Kod Yapısı (Syntax):

```
switch (değişken) {
    case değer1:
        // Değer1'e göre yapılacak işlemler
        break;
    case değer2:
        // Değer2'ye göre yapılacak işlemler
        break;
    default:
        // Hiçbir değer eşleşmezse yapılacak işlemler
}
```

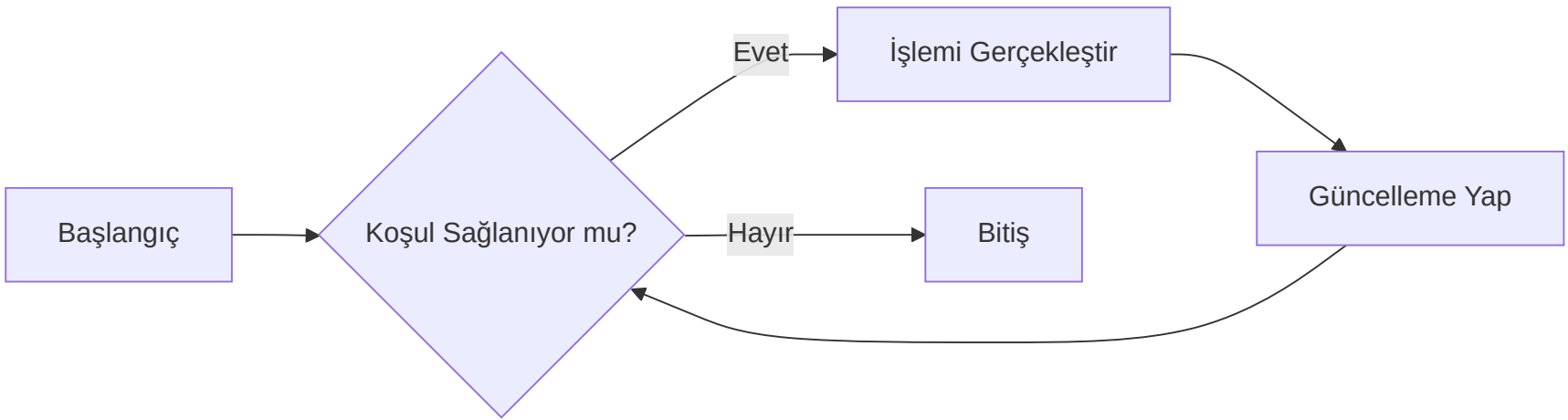
Örnek:

```
char harfNotu = 'B';
switch (harfNotu) {
    case 'A':
        cout << "Mükemmel!" << endl;
        break;
    case 'B':
        cout << "İyi!" << endl;
        break;
    case 'C':
        cout << "Orta!" << endl;
        break;
    default:
        cout << "Geçersiz not!" << endl;
}
```

Döngüler

while Döngüsü

`while` döngüsü, bir koşul doğru olduğu sürece sürekli olarak bir işlemi tekrarlar.



Kod Yapısı (Syntax):

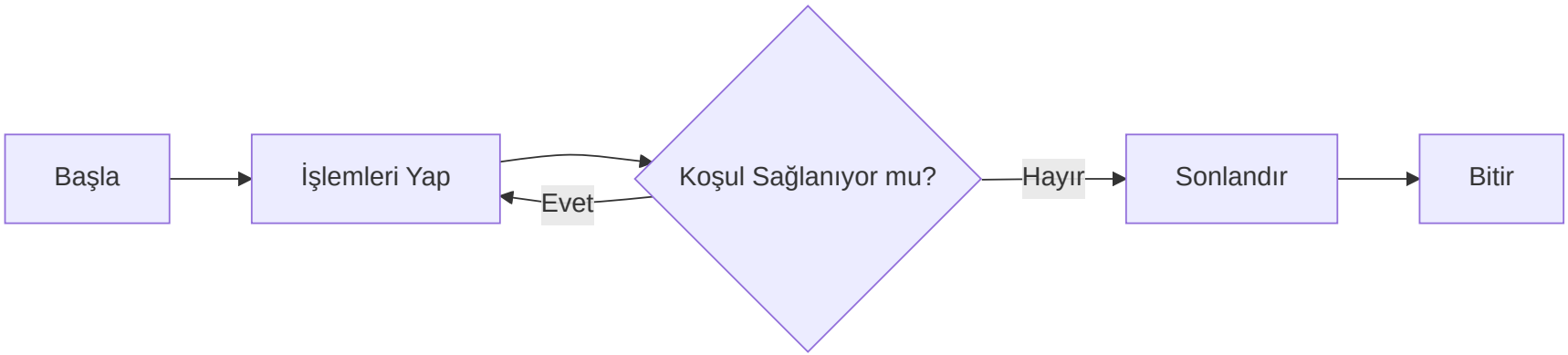
```
Kodu kopyala
while (koşul) {
    // Koşul doğru olduğu sürece yapılacak işlemler
}
```

Örnek:

```
int i = 0;
while (i < 5) {
    cout << "i: " << i << endl;
    i++;
}
```

do-while Döngüsü

`do-while` döngüsü, işlemleri en az bir kere gerçekleştirir ve sonra koşulu kontrol eder.



Kod Yapısı (Syntax):

```
do {
    // Koşul kontrol edilmeden önce yapılacak işlemler
} while (koşul);
```

Örnek:

```
int i = 0;
do {
    cout << "i: " << i << endl;
    i++;
} while (i < 5);
```

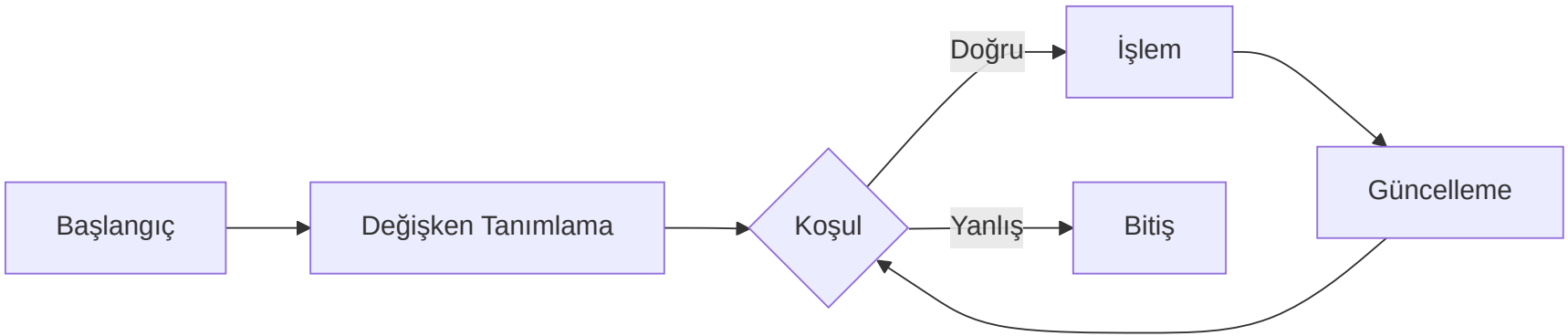
```cpp
#include <iostream>
using namespace std;

int main() {
    int sayi;
    int sayiAdedi = 0;
    int toplam = 0;
    int enBuyuk = 0;
    int enKucuk;
```

```
// Kullanıcıdan pozitif sayılar alın
do {
    cout << "Pozitif bir tam sayı girin (durdurmak için 0 girin): ";
    cin >> sayi;

    if (sayi > 0) {
        sayiAdedi++;
        toplam += sayi;

        if (sayiAdedi == 1) {
            enKucuk = sayi; // İlk sayı en küçük olarak atanır
        }

        if (sayi > enBuyuk) {
            enBuyuk = sayi; // En büyük sayı güncellenir
        }

        if (sayi < enKucuk) {
            enKucuk = sayi; // En küçük sayı güncellenir
        }
    }
} while (sayi != 0); // Kullanıcı 0 girmedikçe döngü devam eder
```

```
    // İstatistikleri yazdır
    if (sayiAdedi > 0) {
        cout << "Toplam sayı adedi: " << sayiAdedi << endl;
        cout << "En büyük sayı: " << enBuyuk << endl;
        cout << "En küçük sayı: " << enKucuk << endl;
        cout << "Ortalama: " << (static_cast<double>(toplam) / sayiAdedi) << endl;
    } else {
        cout << "Hiç pozitif sayı girilmedi." << endl;
    }

    return 0;
}
```

for Döngüsü

`for` döngüsü, belirli bir sayıda tekrarlamak için kullanılır ve genellikle sayaç kontrollüdür.



Kod Yapısı (Syntax):

```
for (başlangıç; koşul; güncelleme) {
    // Koşul doğru olduğu sürece yapılacak işlemler
}
```

Örnek:

```
Kodu kopyala
for (int i = 0; i < 5; i++) {
    cout << "i: " << i << endl;
}
```

break ve continue İfadeleri

break İfadesi

`break` ifadesi, bir döngüyü erken sonlandırmak için kullanılır.

Örnek:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // i 5 olduğunda döngüden çık
    }
    cout << "i: " << i << endl;
}
```

continue İfadesi

`continue` ifadesi, döngünün geri kalan kısmını atlayarak bir sonraki yinelemeye geçmek için kullanılır.

Örnek:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        continue; // i 5 olduğunda bu adımı atla
    }
    cout << "i: " << i << endl;
}
```

Alıştırma 1: Asal Sayı Kontrolü

Soru: Kullanıcıdan bir tam sayı alın ve bu sayının asal olup olmadığını kontrol edin. Asal sayı, yalnızca 1 ve kendisi dışında tam böleni olmayan sayıdır.

Pesudo Kod

```
Başla
    Değişkenler:
        tamSayi, i (tam sayı)
        asal (boolean) = doğru

    Kullanıcıdan bir tam sayı al:
        Yaz "Bir tam sayı girin: "
        Oku tamSayi

    Eğer tamSayi <= 1 ise
        asal = yanlış  // 1 ve daha küçük sayılar asal değildir.
    Aksi durumda
        döngü i = 2'den tamSayi / 2'ye kadar (dahil):
            Eğer tamSayi % i == 0 ise
                asal = yanlış  // Sayı i ile tam bölünüyorsa asal değildir.
                dur
        Son döngü

    Eğer asal ise
        Yaz tamSayi + " bir asal sayıdır."
    Aksi durumda
        Yaz tamSayi + " bir asal sayı değildir."
Sonu
```

```cpp
using namespace std;

int main() {
    int sayi;
    bool asal = true;

    cout << "Bir tam sayı girin: ";
    cin >> sayi;

    if (sayi <= 1) {
        asal = false; // 1 ve daha küçük sayılar asal değildir.
    } else {
        for (int i = 2; i <= sayi / 2; i++) {
            if (sayi % i == 0) {
                asal = false; // Sayı i ile tam bölünüyorsa asal değildir.
                break;
            }
        }
    }

    if (asal) {
        cout << sayi << " bir asal sayıdır." << endl;
    } else {
        cout << sayi << " bir asal sayı değildir." << endl;
    }

```

Alıştırma 2: Fibonacci Dizisi

Fibonacci Dizisi

Fibonacci Dizisi, her bir terimin kendisinden önceki iki terimin toplamı olduğu bir sayı dizisidir. Dizinin ilk iki terimi genellikle 0 ve 1 olarak başlar. Fibonacci dizisi şu şekilde başlar:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Fibonacci Dizisinin Genel Tanımı

Fibonacci dizisinin matematiksel olarak tanımı aşağıdaki gibidir:

- $F(0) = 0$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$ $(n \geq 2)$

Fibonacci Formülü

Fibonacci dizisini hesaplamak için kullanılan formül:

[F(n) = F(n-1) + F(n-2)]

Bu formülde:

- (F(n)): n’inci Fibonacci sayısını temsil eder.
- (F(n-1)): n-1’inci Fibonacci sayısını temsil eder.
- (F(n-2)): n-2’inci Fibonacci sayısını temsil eder.

Örnek Hesaplama

Fibonacci dizisindeki ilk birkaç terimi hesaplayalım:

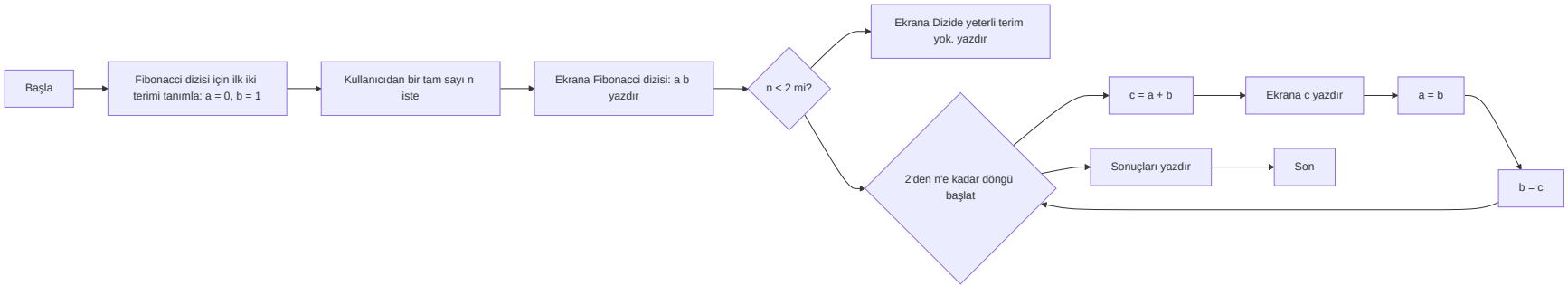
- (F(0) = 0)
- (F(1) = 1)
- (F(2) = F(1) + F(0) = 1 + 0 = 1)
- (F(3) = F(2) + F(1) = 1 + 1 = 2)
- (F(4) = F(3) + F(2) = 2 + 1 = 3)
- (F(5) = F(4) + F(3) = 3 + 2 = 5)
- (F(6) = F(5) + F(4) = 5 + 3 = 8)

Bu şekilde devam ederek Fibonacci dizisinin sonraki terimlerini elde edebilirsiniz.

Soru: Kullanıcıdan bir tam sayı alın ve bu sayıya kadar olan Fibonacci dizisini ekrana yazdırın.

Pesudo Kod:

```
Başla
    Fibonacci dizisi için ilk iki terimi tanımla: a = 0, b = 1
    Kullanıcıdan bir tam sayı n iste
    Ekrana "Fibonacci dizisi: a b" yazdır

    Eğer n < 2 ise
        Ekrana "Dizide yeterli terim yok." yazdır
    Değilse
        2'den n'e kadar döngü başlat
            c = a + b
            Ekrana c yazdır
            a = b
            b = c
        Döngüyü bitir
    Sonuçları yazdır

Son
```


Başla
Fibonacci dizisi için ilk iki terimi tanımla: a = 0, b = 1
Kullanıcıdan bir tam sayı n iste
Ekrana Fibonacci dizisi: a b yazdır
n < 2 mi?
Ekrana Dizide yeterli terim yok. yazdır
2'den n'e kadar döngü başlat
c = a + b
Ekrana c yazdır
a = b
b = c
Sonuçları yazdır
Son

Cevap:

```cpp
using namespace std;

int main() {
    int n, a = 0, b = 1, c;

    cout << "Bir tam sayı girin: ";
    cin >> n;

    cout << "Fibonacci dizisi: " << a << " " << b << " ";
    for (int i = 2; i < n; i++) {
        c = a + b;
        cout << c << " ";
        a = b;
        b = c;
    }

    cout << endl;

    return 0;
}
```

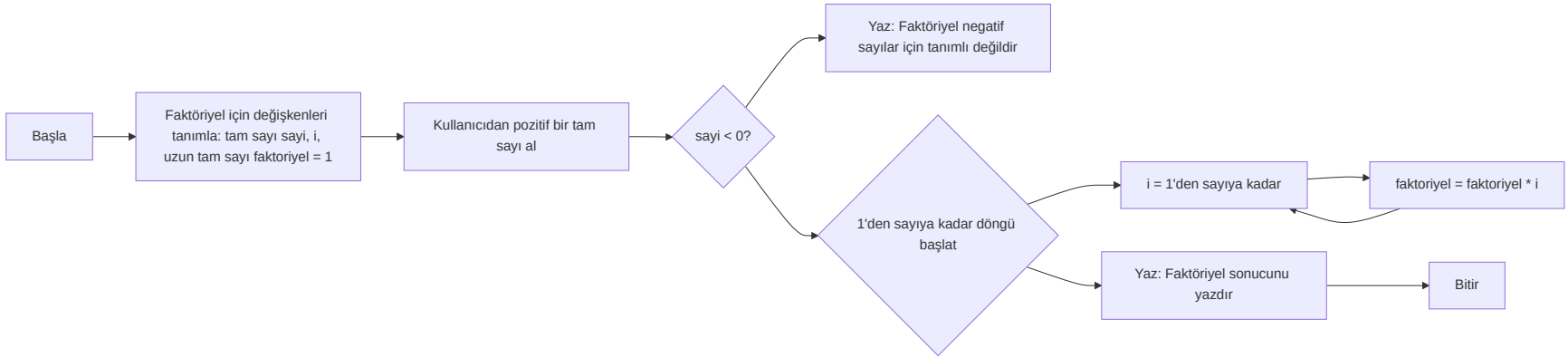
Alıştırma 3: Faktöriyel Hesaplama

Faktöriyel, pozitif bir tam sayının kendisi ve kendisinden küçük pozitif tam sayıların çarpımını ifade eder. Genellikle "n!" şeklinde gösterilir.

Bir pozitif tam sayının faktöriyeli, o sayının ve altındaki tüm pozitif tam sayıların çarpımıdır. Örneğin:

- **0!** = 1 (Faktöriyel 0'ın tanımı gereği 1'dir)
- **1!** = 1
- **2!** = 2 × 1 = 2
- **3!** = 3 × 2 × 1 = 6
- **4!** = 4 × 3 × 2 × 1 = 24
- **5!** = 5 × 4 × 3 × 2 × 1 = 120

Soru: Kullanıcıdan pozitif bir tam sayı alın ve bu sayının faktöriyelini hesaplayın.


Başla
Faktöriyel için değişkenleri tanımla: tam sayı sayi, i, uzun tam sayı faktoriyel = 1
Kullanıcıdan pozitif bir tam sayı al
sayi < 0?
Yaz: Faktöriyel negatif sayılar için tanımlı değildir
1'den sayıya kadar döngü başlat
i = 1'den sayıya kadar
faktoriyel = faktoriyel * i
Yaz: Faktöriyel sonucunu yazdır
Bitir

Pesudo Kod:

```
Başla
    Faktöriyel için değişkenleri tanımla:
        tam sayı sayi
        tam sayı i
        uzun tam sayı faktoriyel = 1

    Kullanıcıdan pozitif bir tam sayı al:
        Yaz("Pozitif bir tam sayı girin:")
        oku(sayi)

    Eğer sayi < 0 ise
        Yaz("Faktöriyel negatif sayılar için tanımlı değildir.")
    Aksi halde
        1'den sayi'ya kadar döngü başlat:
            için i = 1'den sayi'ya kadar
                faktoriyel = faktoriyel * i

        Yaz("Faktöriyel: ", faktoriyel)

Bitir
```

Cevap:

```cpp
#include <iostream>
using namespace std;

int main() {
    int sayi;
    long long faktoriyel = 1;

    cout << "Pozitif bir tam sayı girin: ";
    cin >> sayi;

    if (sayi < 0) {
        cout << "Faktöriyel negatif sayılar için tanımlı değildir." << endl;
    } else {
        for (int i = 1; i <= sayi; i++) {
            faktoriyel *= i; // Faktöriyel hesaplama
        }
        cout << sayi << " sayısının faktöriyeli: " << faktoriyel << endl;
    }

    return 0;
}
```

Alıştırma 4: En Büyük ve En Küçük Sayıyı Bulma

Soru: Kullanıcıdan belirli bir miktarda tam sayı alın ve bu sayılar arasında en büyük ve en küçük olanını bulun.

Pesudo Kod:

```
Başla
    Değişkenler:
        n, sayi, enBuyuk, enKucuk: Tam sayı

    Yaz "Kaç tane sayı gireceksiniz? "
    Oku n

    Yaz "Birinci sayıyı girin: "
    Oku sayi
    enBuyuk ← sayi
    enKucuk ← sayi
```

Pesudo Kod(devam):

```
    Döngü i = 1'den n - 1'e kadar:
        Yaz (i + 1) + ". sayıyı girin: "
        Oku sayi

        Eğer sayi > enBuyuk ise:
            enBuyuk ← sayi
        Son Eğer

        Eğer sayi < enKucuk ise:
            enKucuk ← sayi
        Son Eğer
    Son Döngü

    Yaz "En büyük sayı: ", enBuyuk
    Yaz "En küçük sayı: ", enKucuk
Son
```

Cevap:

```cpp
using namespace std;
int main() {
    int n, sayi, enBuyuk, enKucuk;
    cout << "Kaç tane sayı gireceksiniz? ";
    cin >> n;
    cout << "Birinci sayıyı girin: ";
    cin >> sayi;
    enBuyuk = enKucuk = sayi; // İlk sayıyı hem en büyük hem de en küçük olarak ayarlayın.

    for (int i = 1; i < n; i++) {
        cout << (i + 1) << ". sayıyı girin: ";
        cin >> sayi;

        if (sayi > enBuyuk) {
            enBuyuk = sayi; // En büyük sayıyı güncelle
        }
        if (sayi < enKucuk) {
            enKucuk = sayi; // En küçük sayıyı güncelle
        }
    }

    cout << "En büyük sayı: " << enBuyuk << endl;
    cout << "En küçük sayı: " << enKucuk << endl;

```

Alıştırma 5: Gün Hesaplama

Soru: Kullanıcıdan bir tam sayı alın ve bu sayının haftanın hangi günü temsil ettiğini belirleyin. (1: Pazartesi, 2: Salı, 3: Çarşamba, 4: Perşembe, 5: Cuma, 6: Cumartesi, 7: Pazar). Geçersiz bir sayı girilirse "Geçersiz gün" mesajını verin.

Pesudo Kod:

```
Başla
    Değişkenler:
        gun: Tam sayı

    Yaz "Lütfen bir gün numarası girin (1-7): "
    Oku gun

    Durum gun:
        1:
            Yaz "Gün: Pazartesi"
        2:
            Yaz "Gün: Salı"
        3:
            Yaz "Gün: Çarşamba"
        4:
            Yaz "Gün: Perşembe"
        5:
            Yaz "Gün: Cuma"
        6:
            Yaz "Gün: Cumartesi"
        7:
            Yaz "Gün: Pazar"
        Diğer:
            Yaz "Geçersiz gün"
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int gun;

    cout << "Lütfen bir gün numarası girin (1-7): ";
    cin >> gun;

    switch (gun) {
        case 1:
            cout << "Gün: Pazartesi" << endl;
            break;
        case 2:
            cout << "Gün: Salı" << endl;
            break;
        case 3:
            cout << "Gün: Çarşamba" << endl;
            break;
        case 4:
            cout << "Gün: Perşembe" << endl;
            break;
```

```
        case 5:
            cout << "Gün: Cuma" << endl;
            break;
        case 6:
            cout << "Gün: Cumartesi" << endl;
            break;
        case 7:
            cout << "Gün: Pazar" << endl;
            break;
        default:
            cout << "Geçersiz gün" << endl;
            break;
    }

    return 0;
}
```

Ödev 1: Basamak Toplama

Açıklama: Kullanıcıdan bir tam sayı alın ve bu sayının basamaklarının toplamını hesaplayın. Örneğin, kullanıcı 1234 girdiğinde, 1 + 2 + 3 + 4 = 10 sonucu elde edilmelidir.

Beklenen Sonuç:

- Kullanıcıdan bir tam sayı alın.
- Sayının her bir basamağını ayrıştırın ve toplama işlemi yapın.
- Toplam sonucu ekrana yazdırın.

Yardımcı Kod Örneği:

```cpp
#include <iostream>
using namespace std;

int main() {
    // Kullanıcıdan alınacak sayıyı tanımlayın
    // Döngü ile basamakları toplayın
    // Sonucu yazdırın
}
```

Ödev 2: FizzBuzz Problemi

Açıklama: Kullanıcıdan bir pozitif tam sayı alın ve 1'den bu sayıya kadar olan sayıları yazdırın. Ancak:

- 3'e bölünebilen sayılar için "Fizz",
- 5'e bölünebilen sayılar için "Buzz",
- Hem 3'e hem de 5'e bölünebilen sayılar için "FizzBuzz" yazdırın.

Beklenen Sonuç:

- Kullanıcıdan bir pozitif tam sayı alın.
- 1 ile bu sayı arasındaki her sayıyı kontrol edin ve uygun "Fizz", "Buzz" veya "FizzBuzz" değerlerini yazdırın.

Yardımcı Kod Örneği:

```cpp
#include <iostream>
using namespace std;

int main() {
    // Kullanıcıdan alınacak pozitif tam sayıyı tanımlayın
    // Döngü ile sayıları kontrol edin
    // Uygun sonucu yazdırın
}
```