Programlama Dillerinin Prensipleri

Lab Notları – 4

1. Karar Yapıları

IF Yapıları

Karar yapıları olarak C/C++ ile Java programlama dilleri birbirine yakın ifadeler içerir. Bir programın akışı yukarıdan aşağı doğru ilerler. Bu ilerleyişte bazı satırların bazı koşullarda çalıştırılması istenebilir. Bu durumda kontrol blokları kullanılmalıdır. Karar yapıları, if-if else if else , ternary operator ve switch case şeklinde kullanılabilir. Aşağıda C dilinde basit bir if karar yapısı görülmektedir.

```
#include "stdio.h"
int main(){
   int x;
   printf("Bir sayi girin:");
   scanf("%d",&x);
   if(x % 2 == 0) printf("Girilen sayi cifttir.\n");
   return 0;
}
```

Aynı programı çok fazla ifadeyi değiştirmeden Java'da aşağıdaki gibi yazarız.

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int x;
      System.out.print("Bir sayı girin:");
      x = in.nextInt();
      if(x % 2 == 0) System.out.println("Girilen sayı çifttir.");
   }
```

Peki birden çok kontrol yapılması gerekiyorsa ne yapılmalıdır? Yapılacak işlem if sayılarını çoğaltmak olabilir. Ayrı ayrı if blokları kullanılabileceği gibi içi içe de if blokları kullanılabilir. Burada dikkat edilmesi gereken blokları { } parantezleri ile ayırmaktır. Fakat if içerisinde çalıştırılacak bir ifade varsa parantezlere gerek olmaz.

```
#include "stdio.h"
int main(){
   int x;
   printf("Bir sayi girin:");
   scanf("%d",&x);
   if(x % 2 == 0)
     if(x < 100)
          if(x > 10)
              printf("Girilen sayi 10'dan buyuk 100'den kucuk bir cift sayidir.\n");
   return 0;
}
```

Yukarıdaki aynı durum Java'da da geçerlidir. Yukarıdaki kodu Java'da aşağıdaki gibi genişletirsek yine değişen bir şey olmayacak ve blokları ayıran parantezlere gerek kalmayacaktır. Ama kullanılması da bir hataya sebebiyet vermez. Bu kural aynı şekilde C dili için de geçerlidir.

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int x;
      System.out.print("Bir sayı girin:");
      x = in.nextInt();
      if(x % 2 == 0)
        if(x < 100)
           if(x > 10)
             System.out.println("Girilen sayı 10'dan büyük 100'den küçük ve çift bir sayıdır.");
          else if(x > 5)
             System.out.println("Girilen sayı 5'ten büyük 100'den küçük ve çift bir sayıdır.");
          else
             System.out.println("Girilen sayı 5'ten küçük ve çift bir sayıdır.");
   }
```

Switch-case Yapısı

Kontrol edilecek değerler kesin olarak belli ise switch case yapısı kullanmak daha uygundur. Belirli olmasından kasıt örneğin kullanıcının gireceği bir x değerinin belli sayılardan büyük, belli sayılardan küçük kontrolü if yapısına uygun iken girilecek değer sadece 2 veya 3 yani belli sayıda değer alabiliyorsa switch case yapısı daha uygundur. switch case yapısında switch ifadesinin içindeki değişkenin türü ne ise case ifadeleri o türde kontrol edilmelidir. Örneğin aşağıdaki Java örneğinde kullanıcıdan bir ülke adı girilmesi istenmiş ve ülke adına göre ya yurt içi ya yavru vatan ya da yurt dışı ekrana yazdırılmıştır. Dikkat edileceği üzere kullanıcıdan string türde değer alındığı için case ifadeleri string'leri kontrol etmektedir.

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      String ulke;
      System.out.print("Bir ülke girin:");
      ulke = in.nextLine();
      switch(ulke)
      {
         case "Turkiye":
            System.out.print("Yurt İçi");
            break;
         case "Kibris":
            System.out.print("Yavru Vatan");
            break;
         default:
            System.out.print("Yurt Dışı");
            break;

      }
   }
```

Yukarıdaki default ifadesi switch case yapısının önemli bir unsurudur. default, eğer girilen değer hiçbir case ifadesine uymuyorsa çalışacak olan bloktur. Yazılması zorunlu değildir. switch case çok kullanılmasına karşın düşük seviyeli kontrol ifadesidir. Bundan dolayıdır ki yukarıda Java'da yazılan kodu C dilinde yazılamaz. Çünkü C dilinde switch case yapısında char* kabul edilmemektedir. Hatta daha da ilginci **C dilinde switch case yapısı sadece tam sayıları desteklemektedir**. Yukarıdaki

program C dilinde yazılmak isteniyorsa ya if-else yapısı kullanılmalı ya da değerler tam sayıya dönüştürülüp karşılaştırılmalıdır.

Switch case yapısında bir diğer hayati önem taşıyan durum break ifadelerinin mutlaka konulması gerektiğidir. Konulmaması durumunda C/C++ ve Java herhangi bir hata vermez fakat break konulmayan case ifadesi çalışması durumunda bir alttaki case ifadesini de kontrol etmeden çalıştıracaktır. Örneğin yukarıdaki kod bloğunda case "Turkiye" kontrol bloğundaki break silinirse Turkiye girildiğinde ekrana yurt içi ve yavru vatan ifadelerinin her ikisi de yazacaktır.

Erişilemeyen Satır Durumu

Java gibi yüksek seviyeli bir dil erişilemeyen satıra izin vermez. Erişilemeyen satır hangi koşulda olursa olsun çalıştırılamayacak satırdır. Dolayısıyla yazılmasının bir anlamı yoktur. Örneğin aşağıdaki kodda return altında break kullanılmıştır. Bu satıra hiçbir durumda erişilemez.

```
public class Sayi {
   public static int deger;
   public Sayi(int dgr){
      deger=dgr;
   }
   public int DegerAta(Double yeniDeger){
     String dgr = Double.toString(yeniDeger);
     String ondalik = dgr.substring(dgr.indexOf('.')+1,dgr.length());
     int ondalikKismi = Integer.parseInt(ondalik);
     switch(ondalikKismi)
     {
        case 0: // ondalık kısmı yoktur
           deger = yeniDeger.intValue();
           return deger;
           break; // Erişilemeyen satır
     }
     return 0;
   }
}
```

Fakat aynı durumda C dili hata vermez. Örneğin aşağıdaki kod derlenip çalışacaktır. Fakat bu şekilde bir kod yazımı anlamsız olacağı için kullanılmamalıdır.

```
#include "stdio.h"

int main(){
   int turId;
   printf("Tur girin:");
   scanf("%d",&turId);
   switch (turId) {
      case 1:
         printf("Yonetici");
         return 1;
         break;
      case 2:
         printf("Akademisyen");
         return 2;
```

```
        break;
      case 3:
        printf("Ogrenci");
        return 3;
        break;
   }
         return 0;
}
```

Kontrol bloklarında && || ve ! ifadeleri kullanılabilir. Bunların anlamı && ifadesi ve, || ifadesi veya, ! ifadesi ise değili gösterir. && ifadesinde if bloğunun çalışması için kontrollerden her ikisinin de doğru olması gerekir.

```
#include "stdio.h"

int main(){
        // Girilen sayının pozitif çift sayı olduğunun kontrolü
        int sayi;
        printf("Bir sayi Girin:");
        scanf("%d",&sayi);
        if(sayi % 2 == 0 && sayi >= 0) printf("Girilen sayi pozitif cift sayidir.");
        else printf("Girilen sayi pozitif cift sayi degildir.");
        return 0;
}
```

|| ifadesinde kontrollerden birinin doğru olması if bloğunun çalışmasını sağlar.

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("x:");
      int x = in.nextInt();
      System.out.print("y:");
      int y = in.nextInt();
      if(x % 2 == 0 || y % 2 == 0) System.out.println("x * y çifttir");
      else System.out.println("x * y tektir");
   }
```

Kontrol ifadeleri boolean türden kontrol yapar ve sonuç true ise çalışır false ise çalışmaz. C dilinde 0 değeri false diğer bütün değerler true olarak kabul edilir. Örneğin aşağıdaki C programı ekrana Merhaba yazacaktır.

```
int main(){
        if(1){
                if(135) printf("Merhaba!");
        }
}
```

Fakat aynı şekilde kullanım Java'da derlenme anında hata verecektir. C boolean türü olmadığı için ekrana yazdırılamazken Java'da aşağıdaki gibi bir kullanımda ifadenin doğru olup olmamasına bağlı olarak ekrana true ya da false yazacaktır.

```
 public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("x:");
      int x = in.nextInt();
```

```
        System.out.print("y:");
        int y = in.nextInt();
           System.out.print( x > y );     // Ekrana true ya da false yazar.
        }
```

Aşağıdaki iki kod farklı yazılmalarına rağmen aynı kontrolleri yapıp aynı çıktıları üretirler. C dilinde kullanıcıdan alınacak sayı double ise “%lf” şeklinde alınmalıdır.

```
public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Notunuz:");
        double not = in.nextDouble();
        String harf="";
        if(not >= 90) harf="AA";
        else if(not >= 80) harf = "BA";
        else if(not >= 75) harf = "BB";
        else if(not >= 65) harf = "CB";
        else if(not >= 55) harf = "CC";
        else if(not >= 45) harf = "DC";
        else if(not >= 40) harf = "DD";
        else harf="FF";
        System.out.println("Harf: "+harf);
    }
```

```
int main(){
            double notu;
            char* harf;
            printf("Notunuz (0-100):");
            scanf("%lf",&notu);
            if(notu < 40) harf="FF";
            else if(notu < 45) harf="DD";
            else if(notu < 55) harf="DC";
            else if(notu < 65) harf="CC";
            else if(notu < 75) harf="CB";
            else if(notu < 80) harf="BB";
            else if(notu < 90) harf="BA";
            else harf="AA";
            printf("Harf: %s\n",harf);
            return 0;
}
```

Dizideki eleman ve indeks değerleri verilerek ilgili indekste eleman var mı yok mu kontrolü Java ve C dilinde aşağıdaki gibi yapılmaktadır.

C Dili

```
#include "stdio.h"
#include "stdlib.h"
int main(){
          // 10 uzunluğunda int dizisi
          int *dizi = malloc(10*sizeof(int));
          dizi[0]=25; dizi[1]=32; dizi[2]=40; dizi[3]=3; dizi[4]=11;
          dizi[5]=7; dizi[6]=65; dizi[7]=54; dizi[8]=47; dizi[9]=70;
          int sayi,indeks;
```

Java Dili

```
public static void main(String[] args) {
        // 10 uzunluğunda int dizisi
        Scanner girdi = new Scanner(System.in);
        int []dizi = new int[10];
dizi[0]=25; dizi[1]=32; dizi[2]=40; dizi[3]=3; dizi[4]=11;
dizi[5]=7; dizi[6]=65; dizi[7]=54; dizi[8]=47; dizi[9]=70;
        int sayi,indeks;
        System.out.print("Hangi sayiyi ariyorsunuz:");
```

printf("Hangi sayiyi ariyorsunuz:");
scanf("%d",&sayi);
printf("Sayiyi hangi indekste ariyorsunuz:");
scanf("%d",&indeks);
if(dizi[indeks] == sayi) printf("Sayi var");
else printf("Sayi yok.");
free(dizi);
return 0;
}
	sayi = girdi.nextInt();
	System.out.print("Sayiyi hangi indekste ariyorsunuz:");
	indeks = girdi.nextInt();
	if(dizi[indeks] == sayi) System.out.println("Sayi var");
	else System.out.println("Sayi yok.");
	}

Üçlü Operatör (? :)

Üçlü operatör tek satırda kontrol ve sonucu yazabilmemizi sağlar.
Kontrol Durumu ? Doğru ise çalışır : Doğru değil ise çalışır;

Örneğin aşağıda üçlü operatör ve if kontrollü iki örnek verilmiştir. Fakat yaptıkları iş aynıdır. Java ve C dillerinde kullanım ayınıdır.

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("Sayı:");
      int sayi = in.nextInt();
      String sonuc = (sayi % 2 == 0 ? "Sayı çifttir." : "Sayı tektir.");
      System.out.println(sonuc);
   }
```

```
public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("Sayı:");
      int sayi = in.nextInt();
      if(!(sayi % 2 == 0)) System.out.println("Sayı tektir.");
      else System.out.println("Sayı çifttir.");
   }
```

C dilindeki Karşılığı

```
int main(){
   printf("Sayi:");
   int sayi;
   scanf("%d",&sayi);
   char* sonuc;
   sonuc = (sayi % 2 == 0 ? "Sayi Cifttir" : "Sayi Tektir");
   printf("%s",sonuc);
   return 0;
}
```

Hazırlayan
Yrd. Doç. Dr. M. Fatih ADAK