Programlama Dillerinin Prensipleri
Lab Notları – 2

Veri Türleri

Karakter Tipi (char): C/C++ dillerinde karakter tek bir byte ile ifade edilir. Bir byte 8 bit olduğu için, en fazla 256 karakter ifade edilebilir. Bunlar ASCII kodu olarak ta bilinirler. Fakat Java'da Unicode kullanılmaktadır. Bir karakter için 2 byte ayrılır. Bu da 65536 karakter yazılabilir anlamına gelir. Böylelikle latin harflerinin dışında birçok farklı karakter yapıları da sığdırılabilmektedir.

C ve Java'da \ karateri çıkış karakteri olarak kullanılır. Özel bir karakterdir.

\n Yeni satır
\b Bir karakter geri
\t Tab
\' Tek karakter koymak için
\" Çift karakter koymak için

```
#include "stdio.h"
int main(){
        char c='\b';
        printf("Merhaba%c%c",c,c);
        getch();
        return 0;
}
```

Yukarıdaki kod bloğunda c içerisinde bir karakter geri ifadesi tutulduğu için ekrana Merhaba yazdığında imleç b'nin üzerinde yanıp sönecektir.
Aşağıdaki özel karakterler C/C++'ta bulunmasına rağmen Java'da yoktur.
\a ses çıkarır
\? Soru işareti
\v Dikey tab

Java programlama dilinde karakteri ifade etmek için farklı özel bir yolu vardır. Karakteri hexadecimal değerini yazdırarak ta ekrana çıkartabilirsiniz.

```
public class IlkProje {
   public static void main(String[] args) {
      char a='\u0391';
      System.out.println(a); // Ekrana A harfini yazar.
   }
}
```

Boolean Tipi: Java'da boolean olarak tanımlanan doğru ve yanlış veri türü bellekte 1 bit yer kaplamaktadır, standart C programlama dilinde ise bu türe yer verilmemiştir. C dilinin doğru ve yanlış durumlara bakış açısı biraz farklıdır. Sıfır değeri yanlış kabul edilip bu değer dışındaki bütün değerler doğru olarak kabul edilir. Örneğin aşağıdaki kod parçasında ekrana Sakarya yazacaktır.

```
#include "stdio.h"
int main(){
        if(200) printf("Sakarya");
        else printf("Ankara");
        return 0;
}
```

Yukarıdaki ifade biraz daha iyileştirilirse aşağıdaki gibi tanımlanır ve Java benzeri bir doğru yanlış veri türü elde edilmiş olur. Yapılan şey false ve true iseminde iki kelimenin 0 ve 1 ile ilişkilendirilmesi ve bool ismi ile ifade edilmesidir.

```
#include "stdio.h"
typedef enum {false, true} bool;
int main(){
        bool x=true;
        if(x) printf("Sakarya");
        else printf("Ankara");
        return 0;
}
```

```
#include "stdio.h"
typedef enum {false, true} bool;
int main(){
        bool x=true;
        if(x == true) printf("Sakarya");
        else printf("Ankara");
        return 0;
}
```

Java'da int, float long, double gibi bütün ilkel türlerin boyutları her platformda **sabittir**. Bu taşınabilirliğin getirmiş olduğu bir zorunluluktur. Bundan dolayıdır ki Java'da sizeof operatörü yoktur. Fakat C dilinde durum bu şekilde değildir. Mimariden mimariye ilkel türlerin kaplamış oldukları alanda farklılıklar olabilir. Aşağıdaki kod çalıştırıldığında ekrana 4 yazacaktır. Bu int ilkel türünün bellekte 4 byte kapladığı anlamına gelir. X değişkenine atanan sayının büyüklüğü ile bellekte kapladığı yer arasında bir bağlantı yoktur. Örneğin kodun ikinci kısmında ekrana tekrar 4 byte yazacaktır. Eğer 4 byte'a sığmayacak bir sayı kullanılmak isteniyorsa örneğin double türü düşünülebilir. Aşağıda sizeof'un neden bir fonksiyon değil de operatör olarak ifade edildiği sorulursa aşağıdaki yazılış şeklinden anlaşılabilir.

```
#include "stdio.h"
int main(){
   int x=100;
   printf("%d",sizeof x);
   return 0;
}
```

```
#include "stdio.h"
int main(){
   int x=1000000000;
   printf("%d",sizeof x);
   return 0;
}
```

C dilinde ilkel türler kategori olarak ikiye ayrılırlar, kayan noktalı (ondalık) ve tamsayı olan türler. char, short, int ve long tamsayı türlerine girer. float, double ve long double ise kayan noktalı türlere girer.

Double ile float arasındaki farka bakıldığında Java ve C dilleri için söylenebilecek şey, double türünün, float türüne göre ondalık kısmının daha fazla olduğudur. Aşağıdaki örnek kod C dilinde yazılmış ve double ile float ayrı ayrı kullanılmıştır. Oluşan ekran çıktılarında double'ın daha doğru bir ondalık kısım gösterdiği görülmüştür. Aynı durum Java için de geçerlidir.

```
#include "stdio.h"
int main(){
        float x=10;
        double a=10;
        float y=3;
        double b=3;
        float z = x/y;
        double c = a/b;
        printf("float: %.10f\n",z);
        printf("double: %.10lf\n",c);
```

```
public static void main(String[] args) {
    float x=10;
        double a=10;
        float y=3;
        double b=3;
        float z = x/y;
        double c = a/b;

    System.out.println("float:"+z);
    System.out.println("double:"+c);
```

return 0; }	}
Ekran Çıktısı: float: 3.3333332539 double: 3.3333333333	Ekran Çıktısı: float:3.3333333 double:3.3333333333333335

Yine aynı sebeplerden aşağıdaki karşılaştırma hem Java hem de C dilinde false değerini döndürecektir.

int main(){ float x=0.1; double y=0.1; if(x == y) printf("x ve y esit"); else printf("Esit degil"); return 0; } // False değerini döndürür.	public static void main(String[] args) { float x=0.1f; double y=0.1; if(x == y) System.out.print("x ve y eşit."); else System.out.print("Eşit değil."); } **// Javada ondalık sayılar varsayılan olarak** **// double olduğu için float olarak tanımlak** **// sonuna f getirmekle mümkündür.**

Tür dönüşümlere bakıldığında, C ve Java'da da küçük veri türünden büyük veri türüne dönüştürüldüğünde bir sıkıntı oluşmamaktadır.

```
public static void main(String[] args) {
    int x=100;
    double a=x;
    System.out.println(a);
  }
```

```
int main(){
  int x=100;
  double a=x;
  printf("%lf",a);
  return 0;
}
```

Sıkıntı **büyük veri türünden küçüğüne** dönüştürüldüğünde ortaya çıkmaktadır. C dili esnekliği gereği her hangi bir derlenme hatası vermez. Fakat dönüştürülen değer boyutu daha küçük olan veri türüne sığmayacaksa veri kaybı olur. Örneğin aşağıdaki C kodunda ondalık değer tamsayıya dönüştürülmüş ve ondalık kısmı kaybolmuştur.

```
int main(){
  double x=100.35;
  int a=x;
  printf("%d",a);
  return 0;
}
```

Fakat aynı dönüşüme Java izin vermez ve derlenme anında hata verir. Hatadan kurtulmak için atamanın başına (int) getirilmelidir. Bu Java derleyicisine veri kaybının farkındayım mesajını vermektir. Fakat yine veri kaybının önüne geçilemez.

```
public static void main(String[] args) {
    double x=100.45;
    int a=(int)x;
    System.out.println(a);
  }
```

Java'da türler küçük harf ile başlıyorsa ilkel tür büyük harf ile başlıyorsa o ilkel türün sınıfı olduğunu gösterir. Örnek: Double , double gibi. Ek özellikler kullanılmak isteniyorsa sınıf olanı kullanılmalıdır.

Sabitler

Bazen program yazılırken bazı değerlerin programın sonuna kadar sabit kalması istenebilir. Örneğin pi sayısı veya kat sayılar gibi. Aşağıdaki kod incelendiğinde 9.81'in aslında orada bir sabit olduğu ve değişmemesi gerektiği görülecektir. Fakat kodu analiz eden bir başka programcı 9.81'in belki de sabit olduğunu anlayamayacaktır.

```
public static void main(String[] args) {
      double kuvvet,kutle=78;
      kuvvet = kutle * 9.81;
      System.out.println(kuvvet);
   }
```

Bunun yerine aşağıdaki gibi kullanılması daha açıklayıcı olacaktır.

```
public static void main(String[] args) {
      final double yercekimi = 9.81;
      double kuvvet,kutle=78;
      kuvvet = kutle * yercekimi;
      System.out.println(kuvvet);
   }
```

Yukarıdaki kodda görüldüğü gibi final bir ifadeyi sabit yapmak için kullanılır. C dilnde de bu özellikler geçerli olup sabit tanımı yapmak için const ifadesi kullanılır.

```
int main(){
   const double pi=3.14;
   double yariCap=5.2;
   printf("Cevre=%.2lf",2*pi*yariCap*yariCap);
   return 0;
}
```

Fakat Java ve C dili arasında sabit tanımlamada önemli bir fark bulunmaktadır. Java'da sabite vereceğiniz değer kullanılmadan önce herhangi bir satır olabilir. Fakat C dilinde sabitin tanımlandığı yerde değerini alması gerekmektedir.

```
public static void main(String[] args) {
      final double yercekimi;
      double kuvvet,kutle=78;
      yercekimi=9.81; // C dilinde bu kullanıma izin verilmez.
      kuvvet = kutle * yercekimi;
      System.out.println(kuvvet);
   }
```

var İfadesi

Java 10 ile desteklenmeye başlayan var ifadesi bir değişkene tür tanımı yapmadan değer atamaya izin verir. Fakat burada atanan değere göre tür belirlenmiş olacaktır. Dolayısıyla aşağıdaki hatalı bir kullanım olur. x'e liste atandıktan sonra x'in türü liste olur ve farklı türde bir değer atamaya izin vermez.

```
var x = new ArrayList<Double>();
x="Merhaba";
```

var ifadesi programcıyı bir fonksiyondan gelen türün ne olduğunu arayıp bulmaktan kurtarır. var ifadesi bir anahtar kelimeden çok ayrılmış tür adıdır. var ifadesinin gerekliliği için aşağıdaki örnek incelenebilir. Liste.get(0) diyerek 0. İndeksteki eleman getirilecek ama türü tam olarak ne olduğu hemen anlaşılmayabilir ve yazı karmaşıklığına sebep olabilir. Bunun yerine var yazıp iş kolaylaşmış olacaktır.

```
ArrayList<ArrayList<Double>> liste = new ArrayList<>();
ArrayList<Double> eleman = new ArrayList<>();
eleman.add(85.78);
liste.add(eleman);
var tmp = liste.get(0);
```

Hazırlayan
Dr. Öğr. Üyesi M. Fatih ADAK