**Unions**

Bir struct yapısı gibi, union da türetilmiş bir veri türüdür, ancak aynı depolama alanını paylaşan üyeler içerir. Bir union, kullanılmayan değişkenlerde depolamayı boşa harcamak yerine alanı paylaşır. Bir union üyeleri herhangi bir veri tipinde olabilir. Bir union depolamak için kullanılan bayt sayısı, en azından en büyük üyeyi tutacak kadar olmalıdır. Çoğu durumda, union'lar iki veya daha fazla veri türü içerir. Bir seferde yalnızca bir üyeye ve dolayısıyla bir veri türüne başvurulabilir. Bir union'daki verilere uygun veri türüyle atıfta bulunulmasını sağlamak yazılımcının sorumluluğundadır.

Union tanımı, struct yapısı tanımıyla aynı biçime sahiptir. Aşağıdaki union tanımı, number değişkeninin üyeleri int x ve double y olan bir union türü olduğunu gösterir.

```
union number {
int x;
double y;
};
```

Union tanımı normalde bir başlığa yerleştirilir ve union türünü kullanan tüm kaynak dosyalara dahil edilir.

Bir union üzerinde gerçekleştirilebilecek işlemler şunlardır:

• bir union'ı aynı tipteki başka bir union'a atama.

• bir union değişkeninin adresinin (&) alınması.

• struct yapı üyesi operatörünü ve struct yapı işaretçisi operatörünü kullanarak union üyelerine erişim.

Union'lar da, struct yapıları gibi == ve != operatörleri kullanılarak karşılaştırılamaz.

Bir bildirimde, bir union, ilk union üyesiyle aynı türde bir değerle başlatılabilir. Örneğin, aşağıdaki ifade, union değişkeni değerinin geçerli bir başlatmasıdır çünkü union bir int ile başlatılır

union number value = { 10 };

Ancak aşağıdaki bildirim, başlatıcı değerinin kayan noktalı kısmını keser (bazı derleyiciler bir Bu konuda uyarı verir):

union number value = { 1.43 };

Şekil 10.5'teki program, union'da saklanan değeri hem bir int hem de bir double olarak görüntülemek için union number (satır 6-9) tipi değişken değerini (satır 13) kullanır. Program çıktısı uygulamaya bağlıdır. Program çıktısı, bir double değerinin dahili temsilinin int'nin temsilinden oldukça farklı olabileceğini gösterir.

```
1  // Fig. 10.5: fig10_05.c
2  // Displaying the value of a union in both member data types
3  #include <stdio.h>
4
5  // number union definition
6  union number {
7     int x;
8     double y;
9  };
```

```c

int main(void)
{
   union number value; // define union variable

   value.x = 100; // put an integer into the union
   printf("%s\n%s\n%s\n  %d\n\n%s\n  %f\n\n\n",
      "Put 100 in the integer member",
      "and print both members.",
      "int:", value.x,
      "double:", value.y);

   value.y = 100.0; // put a double into the same union
   printf("%s\n%s\n%s\n  %d\n\n%s\n  %f\n",
      "Put 100.0 in the floating member",
      "and print both members.",
      "int:", value.x,
      "double:", value.y);
}
```