Programlama Dillerinin Prensipleri

HAFTA 4

TEMEL PROGRAMLAMA KAVRAMLARI

VERİ TİPLERİ VE YAPILARI

DR. OGR. UYESİ DENİZ BALTA

Programlama dillerinin temel elemanları

- Bir veya daha çok bellek hücresinin soyutlaması değişkenler yolu ile sağlanmaktadır.
- Her değişken; isim, adres, değer , tip, yaşam süresi ve kapsam özelliklerine sahiptir.
- Bir programlama dilinde aynı isim farklı kısımlarda farklı adreslerle bağlanabilir. Böyle bir durumda her ikisi de farklı değişkenlermiş gibi işlem görecektir ve birbirinden bağımsız olacaktır.
- Yada bir çok tanımlayıcının aynı adresle ilişkilendirilmesi durumunda **örtüşme (aliasing)** durumu ortaya çıkar. Bu durumda programın okuyucusu farklı isimlerdeki değişkenlerin aynı bellek bölgesindeki değerlerini hatırlamak gereksinimi duyacaktır.
- Burada belleği daha iyi kullanmak amaçlanmıştır.

Programlama dillerinin temel elemanları

- Değişkenin değeri belirli bir adreste belirli bir yönteme göre kodlanarak saklanacaktır ve bu değer çalışma sırasında değişebilir.
- Bir değişken programda kullanılmadan önce tanımlanmalı, yani bir isim verilmeli ve tip bildirimi yapılmış olmalıdır.
- Bir değişken tanımlandıktan ve tip bildirimi yapıldıktan sonra bir bellekle ilişkilendirilir.
- Değişkenin bu bellek birimi ile ilişkili kaldığı süreye değişkenin **yaşam süresi (life time)** denir. Tanımlanan değişken programın hangi deyimlerinde geçerli ise bu alanlara da **değişkenin kapsamı** denir.

Değişkenlerin isim özelliği

- İsim değişkenlerin en temel özelliklerinden biridir.
- İsimler aynı zamanda etiketler, altprogramlar, formal parametreler ve diğer pek çok program yapılarının tanımlanmasında da kullanılmaktadır.
- Dil tanımlamalarında isimler yerine tanımlayıcı (identifier) terimi de kullanılmaktadır.

Değişkenlerin isim özelliği

- ***İsimlerin maksimum uzunluğu:***

Programlama Dili	İzin verilen Maksimum isim uzunluğu
FORTRAN I	maksimum 6
COBOL	30
FORTRAN 90, ANSI C	31
Ada	limit yoktur, ve hepsi anlamlıdır(significant)
Java	limit yoktur, ve hepsi anlamlıdır(significant)
ANSI C	31
C++	limit yoktur fakat konabilir

Değişkenlerin isim özelliği

- ***Büyük küçük harf duyarlılığı :***

- Çoğu programlama dilinde büyük/küçük harf farkı yoktur. Ama bazı dillerde bu ayrım önemlidir.
- Örneğin; C, C++, Java programlama dilleri isimlerde küçük-büyük harf duyarlılığını uygulamaktadır.
- Büyük küçük harf duyarlılığını uygulayan dillerde ortalama2 değişkeni ile ORTALAMA2 değişkeni derleyicide farklı değerlendirecektir.
- Böyle bir dilde aynı isimmiş gibi görünen ama farklı anlamlı olan değişkenler olacağından okunabilirlik olumsuz etkilenecektir.
- Büyük küçük harf duyarlılığını uygulamayan dillerde ise ortalama2 değişkeni ile ORTALAMA2 değişkeni arasında fark yoktur.

Değişkenlerin isim özelliği

- ***Özel kelimeler:***

- Özel kelimeler, bir programlama dilindeki temel yapıların kullandığı kelimelerdir.

Anahtar kelimeler(Keywords):

- Bir anahtar sözcük(keyword) yalnızca belirli bir bağlamlarda(kontekstler)(contexts) özel olan sözcüktür(word).

  Örneğin Fortran dilinde:

  Real VarName (Real arkasından bir ad(name) gelen bir veri tipidir(data type), bu yüzden Real bir anahtar sözcüktür (keyword))

  Real = 3.4 (Real bir değişkendir(variable))

  FORTRAN'da REAL TOPLAM yazımında tanımlama deyimi olarak kullanılmıştır.

  Oysa REAL = 87.6 kullanımında bir değişken adıdır.

Değişkenlerin isim özelliği

Ayrılmış kelimeler(Reserved words)

- Programlama dillerinde bazı kelimeler tanımlayıcı olarak kullanılamaz.
- Örneğin C++ dilindeki do, for , while gibi ve PASCAL'da procedure, begin, end gibi kelimere ayrılmış kelime denir.

Örnek: C dili için isimlendirme kuralları:

İsimler, İngilizce büyük ve küçük harfler, rakamlar ve altçizgi işaretinden oluşabilir

İsmin ilk simgesi bir rakam olamaz.

İsmin en az ilk 31 simgesi anlamlıdır.

İsimlerde büyük-küçük harf ayrımı vardır.

C dilinin sözcükleri (int, main, void, return gibi) isimler olamaz.

Kitaplıklardan alınan fonksiyon isimleri ayrılmış kelime değildir.

Değişken ve fonksiyon isimleri küçük harflerle başlar.

Anlamlı isimler verilmek istendiğinde birden fazla sözcüğe gereksinim duyulursa ismi oluşturan iki sözcük bir altçizgi işaretiyle birleştirilir.

Değişkenlerin tipi

- Bir değişkenin hangi aralıklarda değer alabileceği ve bu değişken üzerinde hangi işlemlerin yapılabileceği o değişkenin tipi ile temsil edilir.

  Örneğin tamsayı (integer) tipi, dile bağımlı olarak belirlenen en küçük ve en büyük değerler arasında tamsayılar içerebilir ve sayısal işlemlerde yer alabilir.

Veri tipleri temel veri tipleri ve türemiş tipler olarak incelenebilir.

- Temel veri tiplerini çoğu programlama dili tanımlamıştır ve bu tipler başka tiplerden oluşmayan basit yapıdadır. Tamsayı, mantıksal, karakter, karakter katarı ve kullanıcı tanımlı karakter katarı veri tiplerini temel veri tipleri gurubunda gösterebiliriz.

- Türemiş tipler, çeşitli veri tiplerinde olabilen bileşenlere sahiptir. Yapısal tipin elemanları, tipin bileşenlerini oluştururur ve her bileşenin, tip ve değer özellikleri bulunmaktadır. (Diziler, Record (kayıt) ve (gösterge) pointer örnek olarak verilebilir).

Değişkenlerin tipi

Tablo Temel C++ değişken tipleri				
	Numerik aralık			Bellek alanı
Keyword	Alt sınır	Üst sınır	Ondalık kısım	byte
char	–128	127	yok	1
short	–32,768	32,767	yok	2
int	–2,147,483,648	2,147,483,647	yok	4
long	–2,147,483,648	2,147,483,647	yok	4
float	3.4 x 10–38	3.4 x 1038	7	4
double	1.7 x 10–308	1.7 x 10308	15	8
long double	3.4 x 10–4932	1.1 x 104932	19	10

Sabitler

- Belirli bir tipteki bir değerin kodlanmış gösterimini içeren ancak programın çalıştırılması sırasında değiştirilemeyen bellek hücrelerine sabit denir.
- Eğer ilgili bellek hücresiyle ilişkilendirildiğinde bir değerle de ilişkilendiriliyorsa bu değişkene isimlendirilmiş sabit denir.
- Eğer sabit bir değer programda birçok kez tekrar ediliyorsa programın okunabilirliğini ve yazılabilirliğini artırmak için isimlendirilmiş sabit kullanılır (3.14159 değeri yerine pi ismi kullanılması).
- Ayrıca n elemanlı bir dizide, özellikle döngüsel yapılarda, birçok kez dizi sınırına başvuru yapılır.

Programlama dilleri işlemcileri

GENEL ÖZELLİKLER

İşlenen sayısı:

- Bir işlemci sonuç üretmek için kaç operand gerektiriyorsa bu isimle sınıflandırılabilir.
- Bir işlemci, alabileceği işlenen sayısına göre tekli (unary), ikili(binary) ve üçlü (ternary) olabilir.
- "-" işlemcisi ve C'de bir değişkenin adresini gösteren "&" işlemcisi tekli işlemcilerdir. Bu işlemci için tek operand yeterlidir. "+" ise ikili işlemcilere bir örnektir.
- Koşullu yapıları üçlü işlemcilere örnek gösterebiliriz.

Programlama dilleri işlemcileri Genel özelliklerine göre

İşlemcinin yeri

- Çoğu işlemci işlenenlerin arasına yazılmakla birlikte, bazı işlemciler, işlenenlerinden önce veya sonra da yazılabilirler.
- İşlemciler bir ifadede, işlenenlerden önce(prefix), işlenenler arasında (infix) ve işlenenlerden sonra (postfix) olmak üzere üç şekilde yer alabilirler.

Programlama dilleri işlemcileri Genel özelliklerine göre

- İşlemcilerin öncelikleri, birden çok işlemcinin aynı anda yer aldığı ifadelerde eğer parantez kullanılmamışsa ifadenin hangi sırada işleneceğini belirler.
- Çoğu emir esaslı programlama dilinde ikili işlemciler infix formundadır. Fakat burada "a+b*c" yada “a+b+c” gibi bir ifadenin değerlendirilmesinde sorun oluşabileceğinden öncelik ve birleşme (associativity) kavramlarının açıklanması gerekmektedir.

Öncelik

Programlama dili tasarlanırken her işlemci için önceden belirlenmiş bir öncelik düzeyi vardır.
Yüksek düzeyde önceliğe sahip bir işlemci, işlenenlerini daha düşük bir düzeydeki bir işlemciden önce alır.

Programlama dilleri işlemcileri Genel özelliklerine göre

Birleşme

- "a+b+c" yada a^b^c gibi bir ifadenin değerlendirilmesinde oluşacak sorunlar için birleşme (associativity) özelliğine ilişkin kuralların belirlenmesi gerekir.
- Eğer bir ifadede aynı öncelik düzeyinde iki yada daha fazla işlemci bulunuyorsa, hangi işlemcinin önce değerlendirileceği dilin birleşme (associativity) kuralları ile belirlenir.
- Programlama dili birleşme özelliği yönüyle, işlemcisini sağ veya sol birleşmeli (associative) olarak tanımlayabilir.
- Bir işlemcinin birden çok kez yer aldığı bir ifade, soldan sağa olarak gruplanırsa, işlemci sol birleşmeli olarak adlandırılır. Eğer sağdan sola gruplandırılarak değerlendirme yapılıyorsa sağ birleşmeli olarak adlandırılır.

Programlama dilleri işlemcileri Niteliklerine göre

Sayısal İşlemciler

sembol	İşlev	formül	sonuç
*	Çarpma	4*2	8
/	Bölme ve tamsayı bölme	64/4	16
%	Modul veya kalan	13%6	1
+	Toplama	12+9	21
-	Çıkarma	80-15	65

Programlama dilleri işlemcileri Niteliklerine göre

İlişkisel İşlemciler

Anlamı	C++	PASCAL	FORTRAN 77	ADA	C
Büyüktür	>	>	.GT.	>	>
Küçüktür	<	<	.LT.	<	<
Eşittir	= =	=	.EQ.	=	= =
Eşit değildir	!=	<>	.NE.	/=	!=
Büyük veya eşititr	>=	>=	.GE.	>=	>=
Küçüktür veya eşititr	<=	<=	.LE.	<=	<=

Programlama dilleri işlemcileri Niteliklerine göre

Mantıksal İşlemciler

- Eğer birden fazla koşul sınanacaksa bunların birleştirilip tek bir koşul durumuna getirilmesi gerekir. Böyle durumlarda birden çok koşulun birleştirilmesi için mantıksal operatörler kullanılır.
- Mantıksal işlemciler, sadece mantıksal (Boolean) işlenenleri alarak mantıksal değerler (0:Yanlış yada 1:Doğru) oluştururlar.
- Mantıksal işlemciler, genellikle AND (ve), OR(veya), NOT (değil) ve XOR(özel veya) gibi işlemleri içerirler.
- Sayısal işlemciler gibi mantıksal işlemciler de hiyerarşik öncelik sırasında değerlendirilirler.
- Mantıksal işlemcilerde öncelik sırası, çoğu dilde NOT, AND ve OR biçimindedir.

İşlemci yükleme

- İşlenenlerin sayısına ve tipine bağlı olarak İşlemcilerin anlamları değişebilir. Buna **işlemci yüklemesi (operator overloading)** adı verilir.

- Örneğin "+" işlemcisi tamsayı ve kayan-noktalı sayılarda toplama anlamıyla kullanılırken, bazı dillerde, karakter katarlarının birleştirilmesi için kullanılır.
- İşlemci “-“ de hem çıkarma işlemi için hem de bir sayının negatif olup olmadığını belirlemek için kullanılabilmektedir.

Atama deyimi

- Emir esaslı programlama dillerinde en temel sıralı işlem deyimi atama (assign) deyimidir.
- Atama deyimi ile sağ tarafın içeriğinin sol tarafa aktarılması amaçlanır.

<hedef_değişken> <atama_işlemcisi> <ifade>

- Atama işaretinin sağ ve sol tarafındaki değişkenlerin tip uyumlu olması incelenmesi gereken bir problemdir (tip uyuşmazlığı, tip dönüşümleri, zorunlu tip dönüşümü).

Atama deyimi

Çoklu hedefli atama:

PL/I:

Sum, total=0 deyimi ile hem sum değişkenine hem de total değişkenine 0 değeri atanacaktır.

C:

Sum=total=0 deyimiyle önce total değişkenine 0 değeri atanmakta ve daha sonra total değişkeninin değeri sum değişkenine atanmaktadır.

Atama deyimi

Koşullu Hedefler:

Bu ifadelerde atama bir koşula göre yapılır. C++ ve Java koşullu hedeflere izin vermektedir.

F ? count1:count2=0

ifadesinde F'de belirtilen koşul geçerli ise count1=0 aksi halde count2=0 olacaktır.

Atama deyimi

Bileşik atama:

- Bir atama işlemcisi ile bir ikili işlemci birleştirilerek birleşik atama işlemcilerini oluştururlar.
- C ve C++'da, çeşitli ikili işlemciler için birleşik atama işlemcileri vardır.
- C'de; "+=", "-=", "*=", "/=", "%=" birleşik işlemcileri tanımlıdır.

Operator	örnek	Eşdeğer ifadesi
+=	puan+=500;	puan = puan +500;
-=	c-=50;	c=c-50;
*=	maas*=1.2;	maas=maas*1.2;
/=	factor/=.50;	factor=factor/.50;
%=	d%=7	d=d%7;

Atama deyimi

Tekli atama işlemcileri

- Bu işlemciler yalnızca tek değişkenlere uygulanırlar.

  Örneğin “+”, i = +1; deyiminde ve j = -i deyiminde tek operatör olarak kullanılmıştır.

  “ +5” ve “-5” deyiminde Tek bir + sayının pozitif yada negatif olduğunu göstermektedir.
- Arttırma (++) ve azaltma (- -) operatörleri de tek bir değişken üzerinde işlem yaparlar ve tamsayı değişkenlerde kullanıldığında bir azaltma ve bir arttırma işlemini gerçekleştirirler.
- Eğer bu operatörler karakter değişkenleri ile kullanılıyorsa; örneğin C dilinde k='A' ise ve k++ deyimi çalıştırılırsa k='B' olur.

VERİ TİPLERİ VE YAPILARI

- Bir verinin bellekte nasıl tutulacağını, değerinin nasıl yorumlanacağını ve veri üzerinde hangi işlemlerin yapılabileceğini belirleyen bilgiye **veri tipi** denir.
- Veri tipi kavramının programlama dillerinin gelişiminde çok önemli bir yeri vardır.
- Programlama dillerinde veri tipleri **basit-temel (*primitive*)** ve **türetilmiş (user defined types)** veri tipi olarak iki gurupta incelenir.
- Bunların arasındaki temel fark, temel veri tiplerinin başka veri tiplerinde oluşmamasıdır.

İlkel veri tipleri

- Bu veri tipleri, dilin tasarımında kararlaştırılmış olup ve dilin kurallarına göre varlığı kesin olan türlerdir.
- Farklı programlama dillerindeki önceden tanımlanan veri türleri birbirlerinden farklı olabilir.

  Örneğin C dilinde de önceden tanımlanmış 11 adet veri türü vardır.

  Bu veri türlerinden 8 tanesi tamsayı türünden (Tamsayı veri türleri-integer types) verileri tutmak için, 3 tanesi gerçek sayı (floating types) türünden verileri tutmak için tasarlanmıştır.

İlkel veri tipleri

- C, C++ ve C++ Builder derleyicilerinde kullanılan veri tipleri arasında bazı farklılıklar olmakla birlikte, kullanılan temel veri tipleri aşağıdaki tabloda verilmiştir.

Tip Adı	Uzunluk	Sınırlar	
		Alt Sınır	Üst Sınır
Enum	16 bit	-32,768	32,767
unsigned int	16 bit	0	65,535
short int	16 bit	-32,768	32,767
İnt	16 bit	-32,768	32,767
unsigned long	32 bit	0	4,294,967,295
Long	32 bit	-2,147,483,648	2,147,483,647
Float	32 bit	3.4×10^{-38}	$3.4 \times 10^{+38}$
Double	64 bit	1.7×10^{-308}	$1.7 \times 10^{+308}$
long double	80 bit	3.4×10^{-4932}	$1.1 \times 10^{+4932}$
unsigned char	8 bits	0	255
Char	8 bits	-128	127

İlkel veri tipleri

- **Sayısal (Numeric) Tipler**

Temel sayısal veri tiplerini; tamsayı veri tipi, kayan noktalı veri tipi ve onlu veri tipi olarak inceleyebiliriz.

- **Integer (Tamsayı)**

T**amsayı** (*integer*) veri tipinde bir tamsayı değer, bellekte en sol bit işaret biti olmak üzere bir dizi ikili (*bit*) ile gösterilir.

C dilinin toplam 4 ayrı tamsayı veri türü ve bunların da her birinin kendi içinde işaretli ve işaretsiz biçimi olmak üzere toplam 8 tamsayı türü vardır.

İşaretli (signed) tamsayı türlerinde pozitif ve negatif tam sayı değerleri tutulabilirken, işaretsiz (unsigned) veri türlerinde negatif tamsayı değerleri tutulamaz.

İlkel veri tipleri

- **Floating point (Kayan Noktalı)**

Reel sayıları modellemek için **Kayan noktalı** (*floating point*) veri tipleri tanımlanır.

Kayan noktalı sayılar, kesirler ve üsler olarak iki bölümde ifade edilirler.

Kayan noktalı sayıların tanımlanmasında duyarlılık (*precision*) ve alan (*range*) terimleri kullanılır.

Burada duyarlılık, değerin kesir bölümünün tamlığını; alan ise, kesirlerin ve üslerin birleşmesini ifade eder.

İlkel veri tipleri

- **Decimal (Onlu)**

Ticari işletmelerde ele alınan problemleri desteklemek için bilgisayarlar decimal veri tipini tanımlamışlardır.

Bu veri tipinde onlu değerler tam olarak saklanabilmekte, üsler bulunmadığı için sınırlı bir değer aralığını göstermektedir.

PL/I dilinde tanımlanmıştır. Desimal tipler bilgisayarda BCD (binary coded decimal) Her basamak için bir byte (sekiz bit) kullanılacağından bellek kullanımı bakımından etkin bir veri tipi değildir.

Javada desteklenmez fakat c# dilinde kullanılır. C# dilinde decimal türünde ondalık kısım double a göre çok daha hassastır.

İlkel veri tipleri

- **Karakter (Character) Tipi**

Hesaplamalarda geçerli olmayan, sadece karakterlik bilgi saklayabilen ve bilgisayarda genel olarak ASCII kodlaması ile saklanan bir veri tipidir.

Bu veri tipi değerlerini (128 tane) ASCII tablosundan alır. Bunlardan 8 tanesi kontrol karakteri, 31 tanesi noktalama işaretleri, 26 tanesi yazılamayan karakterler, diğerleri ise (A_Z, a_z) harfler ve (0_9) rakamlardan oluşmaktadır.

C'de *char* ve *int* veri tipleri dönüşümlü olarak kullanılabilmektedir.

C++'da karakter veri tipi de char anahtar kelimesi ile tanımlanır. C++ derleyicisi 'a' gibi bir karakter sabiti ile karşılaştığında bunu ASCII koduna çevirir.

Javada karakterler Unicode olarak saklanır ve bellekte 2 byte yer kaplar.

İlkel veri tipleri

- **Karakter Katarı**

Bir karakter katarı (*character string*) veri tipi karakter dizisi olarak sunulur.

Temel veri tipi olarak tanımlanmamış dillerde (Pascal, C, C++ ve Ada) ise tek karakterli bir karakter dizisi olarak saklanmaktadır.

ADA dilinde STRING tipi vardır ve CHARACTER elemanlarından oluşan tek boyutlu bir dizidir.

ADA dilindeki NAME1:= NAME1 & NAME1 deyimi bu iki kataın bitiştirilmesi anlamına gelmektedir. Eğer NAME1 katarı "Sakarya" ve NAME2 katarı "Üniversitesi" ise yukarıdaki deyim ADA dili için SakaryaÜniversitesi sonucunu verecektir.

İlkel veri tipleri

- **Karakter Katarı**

JAVA programlama dilinde karakter katarları temel veri tipi olarak desteklenmektedir. String class ile desteklenen tiplerin değerleri sabittir ve StringBuffer class ile desteklenen değerler değiştirilebilir.

C dilinde String türü bulunmamaktadır ve yaklaşım tarzı karakter dizisi şeklindedir.

Diziler C'de ilk değerin adresini gösteren bir gösterici olarak tutulmaktadır.

Türetilmiş veri tipleri

- Programlama dillerinin çoğu, önceden tanımlanmış veri türlerine ek olarak, programcının da yeni türler tanımlanmasına izin vermektedir.
- Programcının tanımlayacağı bir nesne için önceden tanımlanmış veri türleri yetersiz kalıyorsa, programcı kendi veri türünü yaratabilir.
- C dilinde de programcı yeni bir veri türünü derleyiciye tanıtabilir ve tanıttığı veri türünden nesneler tanımlayabilir.
- Türetilmiş (*structured*) tipler ilkel tiplerden oluşur ve bellekte bir dizi yerleşimde saklanırlar.
- Diziler, kayıtlar ve göstergeler türetilmiş veri tiplerini oluşturmaktadır.

Türetilmiş veri tipleri

- **Diziler**

Dizilerde bir elemanın yeri ilk elemana göre belirlenebilir.

Dizilerin veri elemanları bir temel veri tipi, ya da daha önceden tanımlanmış bir veri tipidir.

Dizilerin en önemli özelliği, dizideki bir elemana, tanımlayıcı kullanmadan, elemanın dizideki konumunu belirten bir indis aracılığıyla ulaşılabilmesidir.

Dizilerde adres polinomu

- Alt indis sınırı 1 ve her bir elemanının sözcük uzunluğu c olan bir dizinin A[k]'nci elemanın adresi:

  Adres(A[k])= Adres (A[0]) + k * c

  şeklinde tanımlanır.

  İki boyutlu (*i* sıra ve *j* sütunu olan) ve her satırında n eleman bulunan A[i,j] isimli bir dizi için erişim fonksiyonu:

  Adres([a(i,j)]=adres[0,0]+((i-0)*n)+(j-0))*c

  şeklinde tanımlanır.

Record, Union, Küme türü

- **Record (Kayıt) Tipi**

Altalan olarak isimlendirilen birden fazla ifadenin bulunduğu yapıdır.

Kayıt veri tipi ile bir isim altında farklı tipte birden fazla alan tanımlanabilir.

Yani dizilerde homojen elemanlar bulunurken kayıtlarda heterojen elemanlar vardır.

Dizideki bir elemana indis numarası ile ulaşılırken kayıt veri tipinde alanlara her sahayı gösteren tanımlayıcılarla ulaşılmaktadır

Kayıt içerisinde tanımlanan her alanın birbirinden farklı isimleri vardır. İstenirse bu isimler bu tipin dışında farklı tanımlamalar için de kullanılabilir.

Record, Union, Küme türü

- **Union (Ortaklık) Tipi**

Aynı bellek bölgesinin farklı değişkenler tarafından kullanılmasını sağlayan veri tipidir.

Bu veri tipindeki değişkenlere ortak değişkenler de denir.

Buradaki temel amaç farklı zamanlarda kullanılacak birden fazla değişken için ayrı ayrı yer ayırma zorunluluğunun ortadan kaldırılarak belleğin iyi kullanılmasıdır.

Struct veri tipinde her değişken için bellekte ayrı ayrı yer ayrılır.

Struct yapısında tüm alanlara farklı bilgilerin girilmesi mümkündür.

Ortaklık (union) yapısında ise ilgili bellekte tek bilgi tutulur.

Record, Union, Küme türü

- **Set (Küme) Tipi**

Aynı tipte ve birbirleri ile ilgili bilgilerin oluşturduğu bir guruptur.

Küme veri tipi emir esaslı (*imperative)* dillerden sadece Pascal'da tanımlıdır.

Diğer programlama dillerinin desteklememesinin nedeni set bellekte Word boyutuna göre yer ayırır.

- PASCAL dilinde type kısmından faydalanarak *Set of* anahtar kelimesi ile küme veri tipi oluşturulur.

```
Type
Kodlar  =set of [0..255];
Harf    =set of ['A'..'Z'];
Rakam =set of [0..9];
```

Bellek yönetimi

- Bir programdaki değişkenler, sınıflar, metotlar bellekte tutuldukları yer bakımından 3 farklı bölge bulunmaktadır.
- Buna ek çalışan programın kodlarının bulunduğu ayrı bir kısımda derlenmiş kod olarak vardır.

Statik bellek bölgesi

Çalışma anı yığını

Heap bellek bölgesi

Derlenmiş kod

İşaretçiler

- İşaretçiler değişkenlerin kendisini değil adresini gösterir.
- Bir **işaretçi** bir adrestir ve **işaretçi değişken** ise adreslerin saklandığı yerdir.
- İşaretçi tipindeki değerler, gösterdikleri veri ne olursa olsun sabit bir büyüklüktedirler ve genellikle tek bir bellek yerine sığarlar.
- Liste, ikili ağaç ve dizi gibi veri yapıları işaretçilerle daha kolay kullanılabilir.
- İşaretçilerle Bir fonksiyondan başka birine dizi ve stringlerin aktarılması daha güvenli yapılabilir.
- Ayrıca normalde bir fonksiyondan tek bir değer geri alınabilirken işaretçiler ile birden fazla değer almak mümkündür.
- Bir işaretçi değişken tanımı aşağıdaki gibi tanımlanabilir.

*int *pdp;*

Burada *pdp isimli* değişken * karakteri ile işaretçi değişken olarak tanımlanmıştır. *int* ise değişkenin veri tipini göstermektedir.

İşaretçiler

- İşaretçi değişkenler belleğe dolaylı erişim için kullanılabilmektedir.

- Örneğin **« toplam »** isimli bir işaretçi değişkeninin 1950 değerini içersin ve 1950 olan bellek hücresinde 1250 değeri varsa,
  - "**toplam**" değişkenine normal başvuru *1950* değerini,
  - **dolaylı bir başvuru** ise 1250 değerini verecektir.

İşaretçiler

VOID GÖSTERİCİLER

- C dilinde, türü olmayan bir göstericidir.
- Dolayısıyla yeri geldiğinde bir tamsayıyı gösterebileceği gibi yeri geldiğinde bir ondalık sayıyı da gösterebilir.
- Sadece göstericinin gösterdiği yer kullanılacağı zaman, derleyicinin o anki hangi tür olduğu bilmesi açısından dönüştürme işlemi uygulanmalıdır.

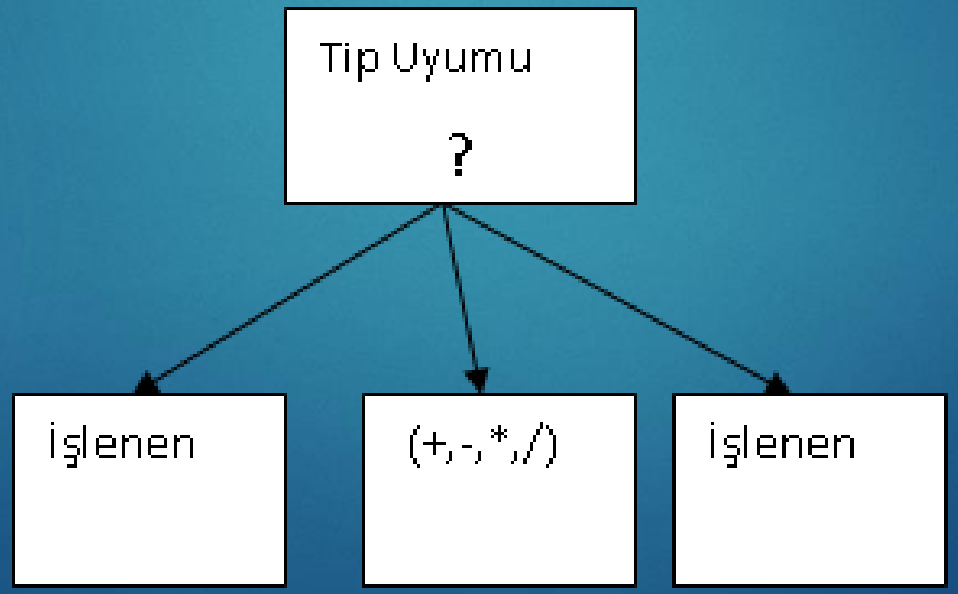
İşaretçiler

SALLANAN GÖSTERİCİLER

- Eğer bir gösterge tipi değişken serbest bırakılmış bir bellek adresini gösteriyorsa yani ilgili adreste geçerli veri bulunmuyorsa bu değişkene **sallanan işaretçi** *(dangling pointer)*.
- Bir işaretçi tipinin tanımlayabileceği veri tipinde bir sınırlama olmaması halinde durağan tip denetimi yapılamadığı için, programların güvenilirliği azalmıştır.
  - İşaretçi değişkenin tanımlandığı ilk dil olan PL/I'da, gösterge değişkenin gösterebileceği elemanların veri tipi sınırlanmamıştır.
  - C'de ise bir işaretçi değişken tanımlanırken, adresini tutabileceği değişken tipi de belirtilmelidir. Bir (int *a) gösteriminde a, sadece int tipinde bir değişkene işaret eder.
- Java programlama dili işaretçi değişken kullanımına izin vermeyerek bu gibi sorunları önlemeye çalışmıştır.

Tip denetimi

- Bir programlama dilinin güvenilirliğinde en önemli etkenlerden biri tip denetimidir.
- Bir işlemcinin işlenenleri birbirleriyle uyumlu tipler olmalıdır. Tip denetimi derleme zamanında ve çalışma zamanında yapılır.



Tip dönüşümleri

- Herhangi bir işlem birden fazla değişken, sabit ve operatör içerebilir.
- İşleme giren değişken ve sabit ifadelerin farklı tiplerden olması doğaldır.
- Bu durumda sonucun hangi tipte olacağını işlem içerisinde değişken ve sabitler belirler.
- Böyle durumlarda işlem içerisinde hafızada en çok yer kaplayan ifadenin veri tipine göre sonucun tipi belirlenir.
- Ancak bazı durumlarda bu hatalı sonuçların oluşmasına sebep olur.

Tip dönüşümleri

- Farklı tipteki sabit veya değişkenler bir ifade içerisinde toplandığında hepsi aynı tipe dönüştürülür.
- Dönüştürme işlemi en yüksek ifadenin tipine göre yapılır.
- İlk olarak bütün **char** ve **short int** değerler **int** veri tipine otomatik olarak yükseltilir.
- Bu adım tamamlandıktan sonra diğer tüm dönüştürmeler aşağıda verilen tip dönüştürme algoritmasına göre adım adım yapılır:

Eğer işlenen **long double** ise ikinci **long double**'a dönüştürülür
Değilse Eğer işlenen **double** ise ikinci **double**'a dönüştürülür
Değilse Eğer işlenen **float** ise ikinci **float**'a dönüştürülür
Değilse Eğer işlenen **unsigned long** ise ikinci **unsigned long**'a dönüştürülür
Değilse Eğer işlenen **long** ise ikinci **long**'a dönüştürülür
Değilse Eğer işlenen **unsigned int** ise ikinci **unsigned int**'a dönüştürülür.

Tip dönüşümleri

- **Karışık tipli ifadelerde** (*mixed mode expression*) bir operatör farklı tiplerde operandlar aldığında eğer farklı operandlar için ayrıca operatör yoksa böyle ifadelerde tip dönüşümlerine ihtiyaç vardır.
- Bir tip dönüşümünde, bir nesne, kendi tipindeki tüm değerleri içermeyen bir tipe dönüştürülüyorsa bu tip dönüşümüne **daralan dönüşüm** denir. Bu dönüşümde hatalar oluşabileceğinden güvenli değildir. Kayan noktalı tipten tamsayıya dönüşüm, daralan dönüşümdür.
- Eğer bir değişkenin kendi tipinin tüm değerlerini içeren bir tipe dönüşümü gerçekleşiyorsa bu dönüşüme genişleyen dönüşüm denir. Daha güvenli bir dönüşümdür. Tamsayı değişkenin kayan noktalı tipe dönüşümü, **genişleyen dönüşümdür.**

Tip dönüşümleri

- Derleyici karışık tipli bir işlemciyle karşılaşırsa, düşük tipteki değişken yüksek tipteki değişkenin tipine dönüştürülür.
- Derleyici tarafından gerçekleştirilen ve zorunlu dönüşüm olarak adlandırılan bu tip dönüşümüne **örtülü** (*implicit*) dönüşüm denir.
- Bu tip dönüşümler derleyicinin tasarımı sırasında belirlenir.

Tip dönüşümleri

Atamalarda tip dönüştürme

- Tip dönüştürme, bir tipteki değişkeni başka bir tipteki değişkene atama durumunda meydana gelir.
- Atama işleminde sağ taraftaki ifade sol taraftaki değişkenin tipine dönüştürülür.

```
#include <stdio.h>
void main()
{
  int x=63;
  float f=27.54;
  char c='A';
  c = x;   // x değişkeni c'ye atanıyor
  x = f;   // f değişkeni x'e atanıyor
  f = c;   // c değişkeni f'ye atanıyor
  f = x;   // x değişkeni f'ye atanıyor
  printf(" %d\n", x );      // sonuç : 27
  printf(" %f\n", f );      // sonuç : 27.0000
  printf(" %d\n", c );      // sonuç : 63 }
```