A

BLM102

PROGRAMLAMA DİLLERİ II

(Özet Bilgi)

Yrd. Doç. Dr. Baha ŞEN

baha.sen@karabuk.edu.tr

KBUZEM
Karabük Üniversitesi
Uzaktan Eğitim Araştırma ve Uygulama Merkezi

1. İŞARETÇİLER

Hemen hemen bütün programlama dillerinin temelinde işaretçi (pointer) veri tipi bulunmaktadır. Birçok dil işaretçi kullanımını kullanıcıya sunmamıştır veya çok sınırlı olarak sunmuştur. Fakat C Programlama Dili'nde işaretçiler yoğun olarak kullanılır. Hatta işaretçi kavramı C dilinin bel kemiğidir. Kavranması biraz güç olan işaretçiler için -latife yapılıp- C kullanıcılarını "işaretçi kullanabilenler ve kullanmayanlar" olmak üzere iki gruba ayıranlar da olmuştur. Özetle, bir C programcısı işaretçi kavramını anlamadan C diline hakim olamaz.

Türkçe yazılan C kitaplarda *pointer* kelimesi yerine aşağıdaki ifadelerden biri karşılaşılabilir:

pointer = işaretçi = işaretçi = gösterge

1.1. Değişken ve Bellek Adresi

Bilgisayarın ana belleği (RAM) sıralı kaydetme gözlerinden oluşmuştur. Her göze bir adres atanmıştır. Bu adreslerin değerleri 0 ila belleğin sahip olduğu üst değere bağlı olarak değişebilir. Örneğin 64 MB bir bellek, 64*1024*1024 = 67108864 adet gözden oluşur. Tabi günümüz bilgisayarları düşünüldüğünde örnekten katlarca fazlası olacaktır. Değişken tiplerinin bellekte işgal ettiği alanın bayt cinsinden uzunluğu sizeof() operatörüyle öğrenilir.

Örnek:

```
#include "stdio.h"
#include "conio.h"
void main()
{
  int m,n;
  n=sizeof(m);
  printf("Size of m= %d",n);
  getch();
}
```

Çıktı: Size of m= 2

Bir programlama dillinde, bir değişken tanımlanıp bir değer atandığında, o değişkenin dört temel özelliği olur:

- değişkenin adı
- değişkenin tipi
- değişkenin sahip olduğu değer (içerik)
- değişkenin bellekteki adresi

Örneğin tam adlı bir tamsayı değişkenini aşağıdaki gibi tanımladığımızı varsayalım:

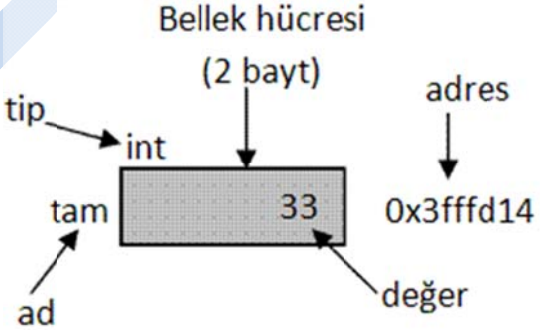
```
int a = 33;
```

Bu değişken için, int tipinde bellekte (genellikle herbiri 1 bayt olan 2 bayt büyüklüğünde) bir hücre ayrılır ve o hücreye 33 sayısı ikilik (binary) sayı sistemindeki karşılığı olan 2 baytlık (16 bitlik):

00000000 00100001

sayısı elektronik olarak yazılır.

tam değişkenine ait dört temel özellik aşağıdaki gibi gösterilebilir:



Şekil 1. Bir değişkene eşlik eden dört temel özellik

Bellek adresleri genellikle onaltılık (hexadecimal) sayı sisteminde ifade edilir. 0x3fffd14 sayısı onluk (decimal) sayı sisteminde 67108116 sayına karışık gelir. Bunun anlamı, tam değişkeni, program çalıştığı sürece, bellekte 67108116. - 67108118. numaralı gözler arasındaki 2 baytlık hücreyi işgal edecek olmasıdır. Şekildeki gösterim, basit ama anlaşılır bir tasvirdir. Gerçekte, int tipindeki tam değişkeninin bellekteki yerleşimi ve içeriği (değeri) şu şekilde olacaktır:

Şekil 2. tam adlı değişkenin bellekteki gerçek konumu ve ikilik düzendeki içeriği

Değişkenin saklı olduğu adres, & karakteri ile tanımlı adres operatörü ile öğrenilebilir. Bu operatör bir değişkenin önüne konursa, o değişkenin içeriği ile değil adresi ile ilgileniliyor anlamına gelir. Aşağıdaki program parçasının:

```
int tam = 33;

printf("icerik: %d\n",tam);

printf("adres : %p\n",&tam);
```

çıktısı:

içerik: 33

adres : 3fffd14

şeklindedir. Burada birinci satır tam değişkeninin içeriği, ikinci ise adresidir. Adres yazdırılırken %p tip belirleyicisinin kullanıldığına dikkat ediniz.

1.2. İşaretçi Nedir?

İşaretçi, bellek alanındaki bir gözün adresinin saklandığı değişkendir. İşaretçilere veriler (yani değişkenlerin içeriği) değil de, o verilerin bellekte saklı olduğu hücrenin başlangıç adresleri atanır. Kısaca işaretçi adres tutan bir değişkendir.

Bir işaretçi, diğer değişkenler gibi, sayısal bir değişkendir. Bu sebeple kullanılmadan önce program içinde bildirilmelidir. İşaretçi tipindeki değişkenler şöyle tanımlanır:

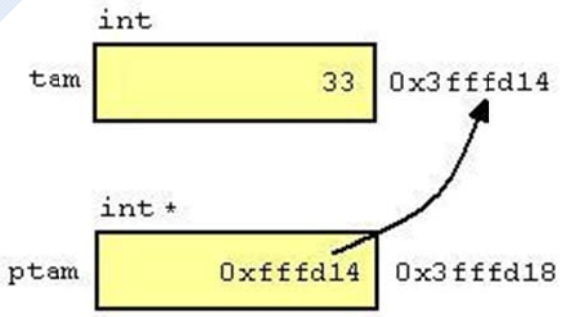
```
tip_adı *işaretçi_adı;
```

Burada tip_adı herhangi bir C tip adı olabilir. Değişkenin önündeki * karakteri yönlendirme (indirection) operatörü olarak adlandırılır ve bu değişkenin veri değil bir adres bilgisi tutacağını işaret eder. Örneğin:

```
char *str; /* tek bir karakter için */
int *a; /* bir tamsayı için */
float *deger, sonuc; /* deger işaretçi tipinde, sonuc
sıradan bir gerçel değişkenler */
```

Yukarıda bildirilen işaretçilerden; str bir karakterin, a bir tamsayının ve deger bir gerçel sayının bellekte saklı olduğu yerlerin adreslerini tutar.

Bir işaretçiye, bir değişkenin adresini atamak için adres operatörünü kullanabiliriz. Örneğin tamsayı tipindeki tam adlı bir değişken ve ptam bir işaretçi olsun. Derleyicide, aşağıdaki gibi bir atama yapıldığında: int *ptam, tam = 33; . . . ptam = &tam; ptam işaretçisinin tam değişkeninin saklandığı adresi tutacaktır. Bu durum aşağıdaki gibi tasvir edilir.



Şekil 3. İşaretçinin bir değişkenin adresini göstermesi

Bu gösterimde, ptam işaretçisinin içeriği tam değişkeninin içeriği (33) değil adresidir (0x3fffd14). Ayrıca, ptam değişkeni, bellekte başka bir hücrede saklandığına ve bu hücrenin int değil int * tipinde bir bölge olduğuna dikkat ediniz.

Örnek:

```
#include "stdio.h"
#include "conio.h"
void main()
{
  int *ptam,tam2, tam1=33;
  ptam=&tam1;
  printf("tam1 degiskeninin degeri= %d\n",tam1);
  printf("tam1 degiskeninin adresi= %p\n",&tam1);

printf("##############################################
########\n");
  printf("ptam degiskeninin isaretcisinin tuttugu
adres= %p\n",ptam);
  printf("ptam isaretcisinin yonlendirdigi adresteki
deger= %d\n",*ptam);

printf("##############################################
########\n");
  getch();
```

```
}
```

tam adlı değişkenin içeriğine ptam işaretçi üzerinde de erişilebilir. Bunun için program içinde ptam değişkeninin önüne yönelendirme operatörü (*) koymak yeterlidir. Yani *ptam, tam değişkeninin adresini değil içeriğini tutar. Buna göre:

```
*ptam = 44;
```

komutuyla, ptam'ın adresini tuttuğu hücreye 44 değeri atanır. Bu durumda tam değişkeninin içeriği dolaylı olarak 44 olmuş olur. **(Lütfen deneyerek görünüz.)**

- *ptam ve tam, tam adlı değişkenin içeriği ile ilgilidir.
- ptam ve &tam, tam adlı değişkenin adresi ile ilgilidir.
- * yönlendirme ve & adres operatörüdür.

1.3. İşaretçi Aritmetiği

İşaretçiler kullanılırken, bazen işaretçinin gösterdiği adres taban alınıp, o adresten önceki veya sonraki adreslere erişilmesi istenebilir. Bu durum, işaretçiler üzerinde, aritmetik işlemcilerin kullanılmasını gerektirir. İşaretçiler üzerinde yalnızca toplama (+), çıkarma (-), bir arttırma (++) ve bir eksiltme (--) operatörleri işlemleri yapılabilir.

Aşağıdaki örnek bu bölümde anlatılanları özetlemektedir. İnceleyiniz.

Örnek:

```
#include "stdio.h"
#include "conio.h"
void main()
{
  int *ptam,tam2, tam1=33;
  ptam=&tam1;
  printf("tam1 degiskeninin degeri= %d\n",tam1);
  printf("tam1 degiskeninin adresi= %p\n",&tam1);

printf("########################################
########\n");
  printf("ptam degiskeninin isaretcisinin tuttugu
adres= %p\n",ptam);
  printf("ptam isaretcisinin yonlendirdigi adresteki
deger= %d\n",*ptam);

printf("########################################
########\n");
  tam2=*ptam;
  printf("tam2 degiskeninin degeri= %d\n",tam2);
  printf("tam2 degiskeninin adresi= %p\n",&tam2);
  printf("tam2 degiskeninin adresi= %p\n",ptam+1);
  getch();
```

```
}
```

1.4. İşaretçi ve Diziler Arasındaki İlişki

C dilinde işaretçiler ve diziler arasında yakın bir ilişki vardır. Bir dizinin adı, dizinin ilk elemanının adresini saklayan bir işaretçidir. Bu yüzden, bir dizinin herhangi bir elemanına işaretçi ile de erişilebilir. (Önceki örnekte de gösterildiği gibi.)Örneğin:

```
int *ptam, tam[5]={1, 2, 3, 4, 5};
ptam=&tam[0];
```

şeklinde bir bildirim yapılsın. Buna göre aşağıda yapılan atama geçerlidir:

`ptam=&tam[0];/* Dizinin birinci elemanının adresi ptam işaretisine atandı*/`

Örnek:

```
#include "stdio.h"
#include "conio.h"
void main()
{
  int *ptam, tam[5]= {1,2,3,4,5};
  ptam=&tam[0];
  for (int i=0; i<5; i++)
    {
      printf("%d. elemanin adresi= %p\t",i+1,ptam+i);
      printf("icerik=%d\n",*(ptam+i));
    }
```

```
  getch();
}
```