Algoritma ve Programlama-1

10. HAFTA
Göstericiler (Pointers), Gösterici Operatörleri, Değişken ve Bellek Adresi, Gösterici Aritmetiği

9. Hafta - Tekrar

- Vize Sınavı Hakkında
  - Soruların Çözülmesi
  - Sonuçların Değerlendirilmesi
  - Analizler
  - Görüşler

Göstericiler (Pointers)

- C bir gösterici dilidir, göstericisiz bir C programı düşünülemez.
- Göstericiler, C öğreniminin bel kemiğini oluşturur.
- Ancak öğrenciler diğer programlama dillerinin çoğunda bulunmayan göstericilere zor ısınırlar ve anlamakta güçlük çekerler.
- Göstericilerin, C eğitiminde kötü bir ünü vardır.
- Kavranması biraz güç olan göstericiler için -latife yapılıp- C kullanıcılarını "gösterici kullanabilenler ve kullanmayanlar" olmak üzere iki gruba ayıranlar da olmuştur.
- Özetle, bir C programcısı gösterici kavramını anlamadan C diline hakim olamaz.
- **pointer = işaretçi = gösterici = gösterge**

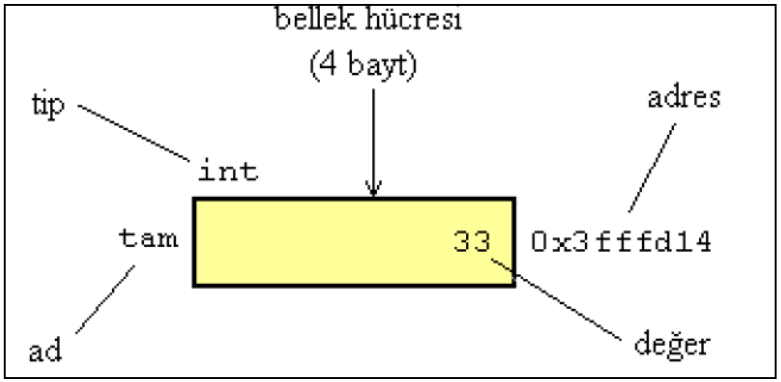
Değişken ve Bellek Adresi

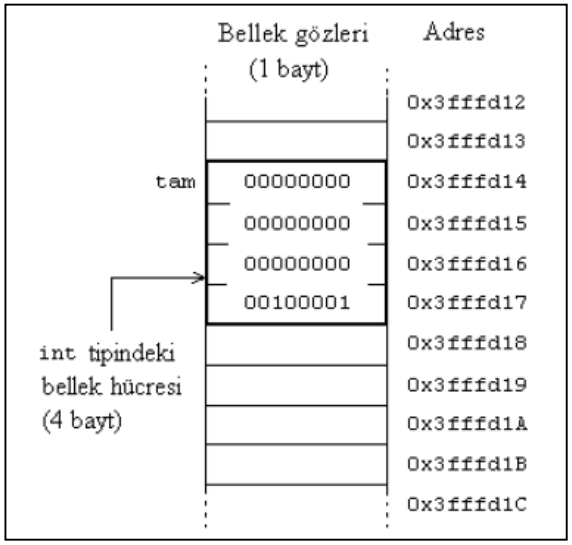
- Bir programlama dillinde, belli bir tipte değişken tanımlanıp ve bir değer atandığında, o değişkene dört temel özellik eşlik eder:
  - 1. Değişkenin adı
  - 2. Değişkenin tipi
  - 3. Değişkenin sahip olduğu değer (içerik)
  - 4. Değişkenin bellekteki adresi

- Örneğin ***tam*** adlı bir tamsayı değişkenini aşağıdaki gibi tanımladığımızı varsayalım:
  - int tam = 33;

- Bu değişken için, int tipinde bellekte bir hücre ayrılır ve o hücreye 33 sayısı ikilik (binary) sayı sitemindeki karşılığı olan 4 baytlık (32 bitlik):
  - 00000000 00000000 00000000 00100001

    sayısı elektronik olarak yazılır.





Göstericiler (Pointers)

- Gösterici, bellek alanındaki bir gözün adresinin saklandığı değişkendir.
- Göstericilere veriler (yani değişkenlerin içeriği) değil de, o verilerin bellekte saklı olduğu hücrenin başlangıç adresleri atanır.
- Kısaca gösterici adres tutan bir değişkendir.
- Göstericiler adres bilgilerini saklamak ve adreslerle ilgili işlemler yapmak için kullanılırlar.

Göstericilere Neden İhtiyaç Duyarız?

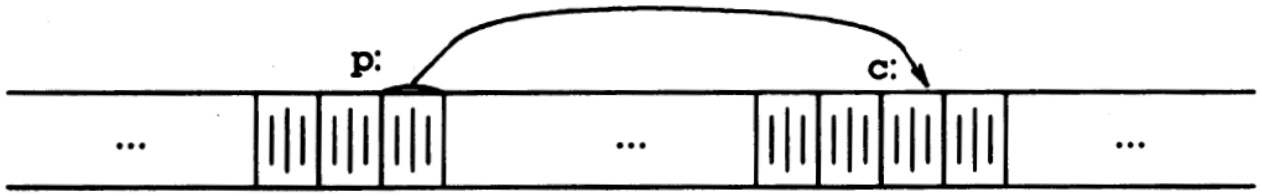
- Normalde bir fonksiyondan tek bir değer geri alınabilir. Göstericiler ile birden fazla değer almak mümkündür.
- Bir fonksiyondan başka birine dizi ve stringlerin aktarılması daha güvenli yapılabilir.
- Dizilerle daha kolay çalışılabilir.
- Liste, ikili ağaç gibi kompleks veri yapıları daha kolay kullanılabilir.
- Bellek kullanımı ve yönetimi daha kolaydır.

Pointers

- A pointer is a variable that contains the address of a variable.
- Pointers are much used in C,
  - Partly ,because they are sometimes the only way to express a computation, and
  - Partly because they usually lead to more compact and efficient code than can be obtained in other ways.
- Pointers have been lumped with the goto statement as a marvelous way to create impossible-to-understand programs.
- This is certainly true when they are used carelessly, and it is easy to create pointers that point somewhere unexpected.
- With discipline, however, pointers can also be used to achieve clarityand simplicity.
- This is the aspect that we will try to illustrate.

Pointers and Adresses

- Let us begin with a simplified picture of how memory is organized.
- A typical machine has an array of consecutively numbered or addressed memory cells that may be manipulated individually or in contiguous groups.
- One common situation is that any byte can be a char, a pair of one byte cells can be treated as a short integer, and four adjacent bytes form a long.
- A pointer is a group of cells (often two or four) that can hold an address.
- So if *c* is a char and *p* is a pointer that points to it, we could represent the situation this way:

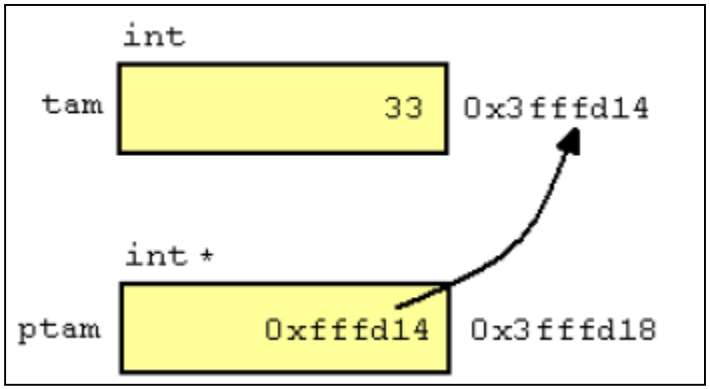


Göstericiler

- Bir gösterici, diğer değişkenler gibi, sayısal bir değişkendir.
- Bu sebeple kullanılmadan önce program içinde bildirilmelidir. Gösterici tipindeki değişkenler şöyle tanımlanır:
  - *tip_adı *gösterici_adı*;
- Burada *tip_adı* herhangi bir C tip adı olabilir.
- Değişkenin önündeki * karakteri içerik/yönlendirme(indirection) ve referanstan ayırma (dereferencing) operatörü olarak adlandırılır ve bu değişkenin veri değil bir adres bilgisi tutacağını işaret eder.
- Örneğin:
  - char *kr; /* tek bir karakter için */
  - int *x; /* bir tamsayı için */
  - float *deger, sonuc; /* deger gösterici tipinde, sonuc sıradan bir gerçel değişkenler */
- Yukarıda bildirilen göstericilerden; kr bir karakterin, x bir tamsayının ve deger bir gerçel sayının bellekte saklı olduğu yerlerin adreslerini tutar.

Göstericiler

- Bir göstericiye, bir değişkenin adresini atamak için adres operatörünü kullanabiliriz.
- Örneğin tamsayı tipindeki tam adlı bir değişken ve ptam bir gösterici olsun.
- Derleyicide, aşağıdaki gibi bir atama yapıldığında:
  - int *ptam, tam = 33;
  - ptam = &tam;
- *ptam* göstericisinin tam değişkeninin saklandığı adresi tutacaktır.



Pointers

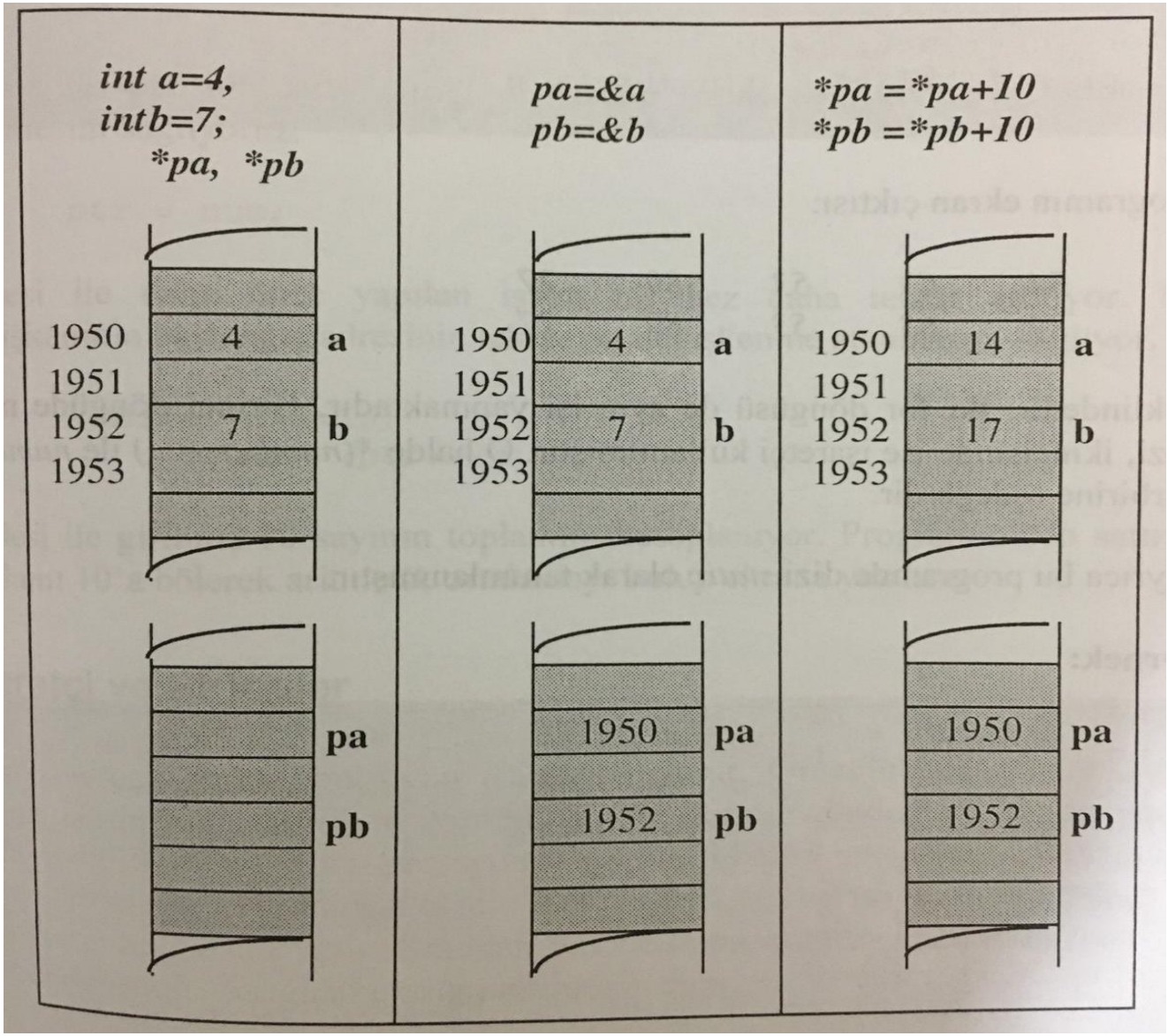
- The unary operator & gives the address of an object, so the following statement assigns the address of c to the variable p, and p is said to "point to" c.
  - p = &c;
- The & operator only applies to objects in memory: variables and array elements.
- It cannot be applied to expressions, constants, or register variables.
- The unary operator * is the *indirection* or *dereferencing* operator; when applied to a pointer, it accesses the object the pointer points to.
- Suppose that x and y are integers and ip is a pointer to int.
- This artificial sequence shows how to declare a pointer and how to use & and *:

```
int x = 1, y = 2, z[10];
int *ip;          /* ip is a pointer to int */

ip = &x;          /* ip now points to x */
y = *ip;          /* y is now 1 */
*ip = 0;          /* x is now 0 */
ip = &z[0];       /* ip now points to z[0] */
```

Gösterici Operatörleri

- *
  - İçerik, Yönlendirme
  - Indirection, Dereference
- &
  - Adres
  - Address, Reference
- [n]
  - İndeks, Index

Göstericiler



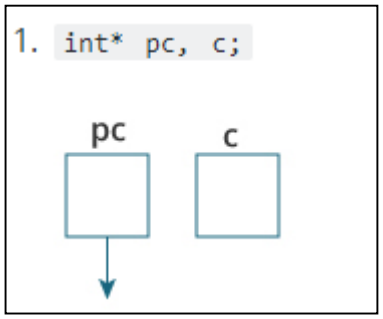
Göstericiler

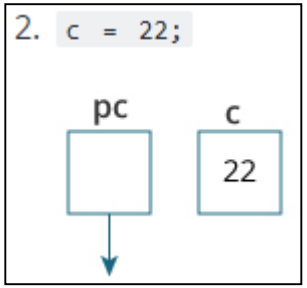
```c
int main()
{
    int* pc, c;

    c = 22;
    printf("Address of c: %u\n", &c);
    printf("Value of c: %d\n\n", c);

    pc = &c;
    printf("Content Address of pointer pc: %u\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc);

    c = 11;
    printf("Content Address of pointer pc: %u\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc);

    *pc = 2;
    printf("Address of c: %u\n", &c);
    printf("Value of c: %d\n\n", c);
    return 0;
}
```

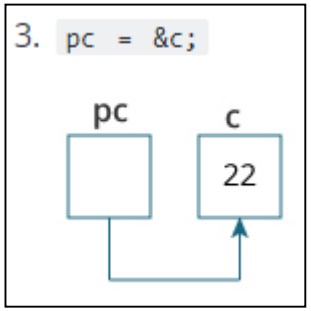
```
Address of c: 6356744
Value of c: 22

Content Address of pointer pc: 6356744
Content of pointer pc: 22

Content Address of pointer pc: 6356744
Content of pointer pc: 11

Address of c: 6356744
Value of c: 2
```

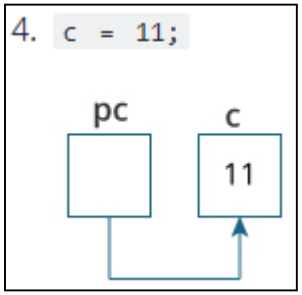
Göstericiler

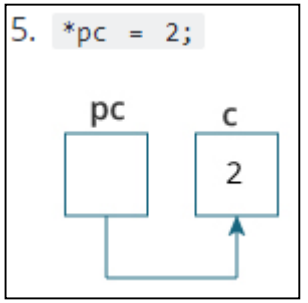
```c
int main()
{
    int* pc, c;

    c = 22;
    printf("Address of c: %u\n", &c);
    printf("Value of c: %d\n\n", c);

    pc = &c;
    printf("Content Address of pointer pc: %u\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc);

    c = 11;
    printf("Content Address of pointer pc: %u\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc);

    *pc = 2;
    printf("Address of c: %u\n", &c);
    printf("Value of c: %d\n\n", c);
    return 0;
}
```



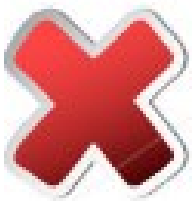








Common mistakes when working with pointers



int c, *pc;

- pc = c;
  - **Wrong!** pc is address whereas, c is not an address.
- pc = &c;
  - **Correct!** pc is an address and, &c is also an address.
- *pc = &c;
  - **Wrong!** *pc is the value pointed by address whereas, &c is an address.
- *pc = c;
  - **Correct!** *pc is the value pointed by address and, c is also a value (not address).

Göstericiler Uygulama

- Bir int değişken ve pointer tanımlayarak, değişken adresini pointer'a atayınız ve tüm değişken içerik ve adreslerini yazdırınız.

- Pointer kullanarak toplama işlemini gerçekleştiriniz.

- İki değişkenin değer değiştirme (swap) işlemini pointer kullanarak gerçekleştiriniz.

Gösterici Aritmetiği

- Göstericiler kullanılırken, bazen göstericinin gösterdiği adres taban alınıp, o adresten önceki veya sonraki adreslere erişilmesi istenebilir.

- Bu durum, göstericiler üzerinde, aritmetik işlemlerinin kullanılmasını gerektirir.

- Göstericiler üzerinde yalnızca toplama (+), çıkarma (-), bir arttırma (++) ve bir eksiltme (--) operatörleri işlemleri yapılabilir.

- Aşağıdaki gibi üç tane gösterici bildirilmiş olsun:
  - char *kar;
  - int *tam;
  - double *ger;

- Bu göstericiler sırasıyla, bir karakter, bir tamsayı ve bir gerçel sayının bellekte saklanacağı adreslerini tutar.

Gösterici Aritmetiği

- Herhangi bir anda, tuttukları adresler de sırasıyla 10000 (0x2710), 20000 (0x4e20) ve 30000 (0x7530) olsun.
- Buna göre aşağıdaki atama işlemlerinin sonucu:
  - kar++;
  - tam++;
  - ger++;

  sırasıyla 10001 (0x2711), 20004 (0x4e24) ve 30008 (0x7538) olur.
- Bir göstericiye ekleme yapıldığında, o anda tuttuğu adres ile eklenen sayı doğrudan toplanmaz.
- Böyle olsaydı, bu atamaların sonuçları sırasıyla 10001, 20001 ve 30001 olurdu.
- Gerçekte, göstericiye bir eklemek, göstericinin gösterdiği yerdeki veriden hemen sonraki verinin adresini hesaplamaktır.

Gösterici Aritmetiği

- Genel olarak, bir göstericiye *n* sayısını eklemek (veya çıkarmak), bellekte gösterdiği veriden sonra (veya önce) gelen *n*. elemanın adresini hesaplamaktır.

- Buna göre aşağıdaki atamalar şöyle yorumlanır.
  - kar++; /* kar = kar + sizeof(char) */ 5
  - tam = tam + 5; /* tam = tam + 5*sizeof(int) */
  - ger = ger - 3; /* ger = ger - 3*sizeof(double) */

Göstericilerin Uzunlukları

- Bir gösterici tanımlamasıyla karşılaşan derleyici –diğer tanımlamalarda yaptığı gibi- bellekte o gösterici için yer tahsis eder.
- Derleyecilerin göstericiler için tahsis ettikleri yerlerin uzunluğu donanım bağımlı olup sistemden sisteme değişebilmektedir.
- Göstericiler uzunluğu türlerinden bağımsızdır.
- Örneğin;
  - char *s;
  - int *p;
  - float *f
- DOS altında s,p ve f göstericilerin hepsi de bellekte 2 byte ya da 4 byte yer kaplarlar.
- Çünkü göstericilerin türü yalnızca gösterdikleri yerle ilişkilidir.

Operatör Öncelikleri

- İndeks Operatörü ([ ])
  - ++ ve -- operatörlerinin indeks operatörü ile kullanılmasında çatışmaya yol açacak bir durum yoktur.
  - İndeks operatörü daha önceliklidir.

- Adres Operatörü (&)
  - ++ ve -- operatörleri adres operatörü ile yan yana kullanılamaz.

Operatör Öncelikleri

- İçerik Operatörü (*)
  - Üç durum oluşabilir.

1. *operand++ / *operand--
   - Önce ++ / -- operatörü, sonra *

```
while(*s++!='\0')   vs   while(*s!='\0')  ++s;
```

2. *++operand / *--operand
   - Önce ++ / --, sonra *

```
char s[20];
char *p;
p=s;
*++p='A'
```

3. ++*operand / --*operand
   - Önce *, sonra ++ /--

```
char ch='a';
char *s;
s=&c;
++*s;
```

Null Pointer

- Bir göstericinin bellekte herhangi bir adresi göstermesi, veya önceden göstermiş olduğu adres iptal edilmesi istenirse NULL sabiti kullanılır.

- Bu sabit derleyicide ASCII karakter tablosunun ilk karakteridir ve '\0' ile sembolize edilir.

```
int *ptr, a = 12;
.
.
ptr = &a;      /* ptr bellekte a değişkenin saklandığı yeri gösteriyor  */
.
.
ptr = NULL;    /* ptr bellekte hiç bir hücreyi göstermiyor */
*ptr = 8       /* hata! NULL göstericinin gösterdiği yere bir değer atanamaz */
```

void Gösterici (Generic Pointer)

- void göstericiler herhangi bir veri tipine ait olmayan göstericilerdir.
- Bu özelliğinden dolayı, void gösterici genel gösterici (generic pointer) olarak da adlandırılır.
  - void *p;
- void göstericiler yalnızca adres saklamak için kullanılır.
- Bu yüzden diğer göstericiler arasında atama işlemlerinde kullanılabilir.
- Örneğin aşağıdaki atamada derleyici bir uyarı veya hata mesajı vermez:

```
void *v;
char *c;
.
.
.
v = c;    /* sorun yok !*/
```

Generic Pointer

```c
#include <stdio.h>

int main()
{
  char    kar = 'a';
  int     tam = 66;
  double  ger = 1.2;
  void   *veri;

  veri = &kar;
  printf("veri -> kar: veri  %c  karakter degerini
gosteriyor\n", *(char *) veri);

  veri = &tam;
  printf("veri -> tam: simdi veri  %d  tamsayi degerini
gosteriyor\n", *(int *) veri);

  veri = &ger;
  printf("veri -> ger: simdi de veri  %lf  gercel sayi
degerini gosteriyor\n", *(double *) veri);

  return 0;
}
```

void Gösterici

- Belirli bir türe sahip olmadıkları için void göstericiler * ve [n] gösterici operatörleri ile kullanılamaz.
- Çünkü bu operatörler değere erişmek için tür bilgisine de ihtiyaç duyarlar.
- Bir void gösterici artırılamaz ya da eksiltilemez.
- Çünkü bir göstericiyi 1 arttırdığımızda içerisindeki adres göstericinin türünün uzunluğu kadar artmaktadır.
- Void göstericiler adresleri geçici olarak saklamak amacıyla kullanılırlar.
- Diğer göstericiler arasındaki atama işlemlerinde uyarı ya da hata oluşturmadıklarından dolayı, türden bağımsız adres işlemlerinin yapıldığı fonksiyonlarda parametre değişkeni ya da geri dönüş değeri olabilirler.

Sabit Göstericiler (Const Pointers)

- const tür belirleyicisi ile bildirilmiş olan göstericilere sabit göstericiler denir.

- const anahtar sözcüğünün bildirimdeki yerine bağlı olarak sabit göstericilerin anlamı değişmektedir.
  - Gösterdikleri yer sabit olan sabit göstericiler
    - int *const ptr;
  - Kendisi sabit olan sabit göstericiler
    - const int* ptr
  - Hem kendisi hem de gösterdiği yer sabit olan sabit göstericiler
    - const int* const ptr

Fonksiyon Göstericileri

- Program kodları da değişkenler gibi bellekte yer kapladığına göre onlarında adresinden söz edilebilir.

- Bir fonksiyonun adresi, o fonksiyona ilişkin kodun bellekteki başlangıç adresidir.

- Genel olarak adresleri veri adresleri ve kod adresleri olmak üzere iki bölüme ayırabiliriz.
  - Veri Adresleri – int,char vb. gibi değişken adresleri
  - Kod Adresleri – fonksiyonların başlangıç adresleri

Fonksiyon Göstericilerinin Bildirimleri

- Bir fonksiyon göstericisi, gösterdiği fonksiyonun parametre ve geri dönüş değerlerinin türü belirtilerek bildirilir.
  - <geri dönüş değerinin türü> (* gösterici ismi) (parametre türleri);
  - int (*p) (void);
  - int (*s) (int,int);

```
int topla(int a, int b)
{
    return a + b;
}

void main(void)
{
    int (*p) (int, int);

    p = topla;
    ...
}
```

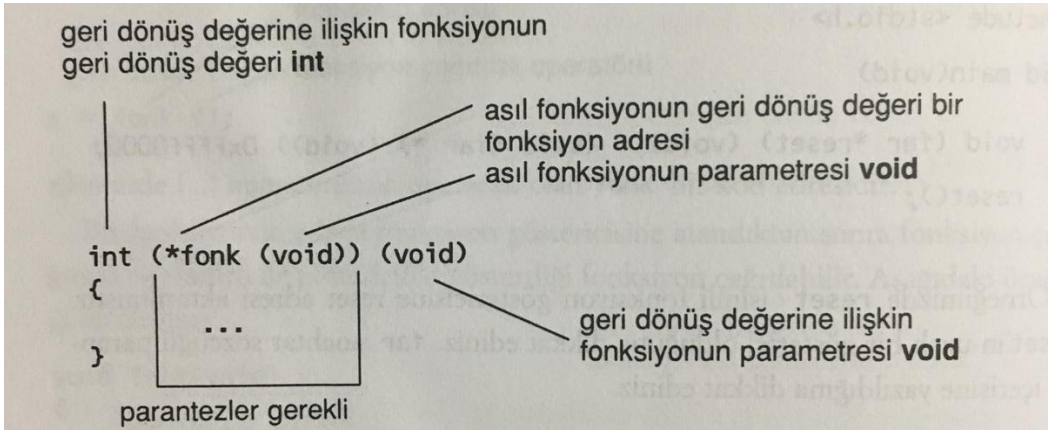
Fonksiyon Göstericiler

- Fonksiyon göstericileri, fonksiyon parametresi ve geri dönüş değeri olarak kullanılabilirler.

- double fonk (int x,int y, int (*p) (int, int))
  - Biçiminde tanımlanmış fonksiyonun 3. parametresi geri dönüş değeri int, parametreleri de int,int biçiminde olan bir fonksiyonun adresidir.

- Aşağıda geri dönüş değeri int (*p) (void) biçiminde bir fonksiyon olan fonksiyona örnek verilmiştir.

- Fonk isimli fonksiyonun parametresi void ve geri dönüş değeri «parametresi void geri dönüş değeri int olan» bir fonksiyonun adresidir.



HELP!

Önümüzdeki Hafta

- Göstericiler ve Diziler
- Göstericiler ve Fonksiyonlar
- Karakter Göstericiler

Kaynakça

1. Aslan, K., A'dan Z'ye C Kılavuzu, Pusula Yayınları.
2. Bişkin F., C Programlama Diline Giriş Notları.
3. Aksoy,M.S., Akgöbek, Ö., C Programlama ve Programlama Sanatı, Beta Yayınları.
4. Kernighan, K.W., Ritchie, D.M., The C Programming Language, Prentice Hall Software Series.
5. https://www.programiz.com/c-programming/c-pointers