

**MALATYA  
TURGUT ÖZAL ÜNİVERSİTESİ  
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
ALGORİTMA VE PROGRAMLAMA 2**

**Gösterge (Pointer) Kavramı**

## Pointers (İşaretçiler)

- Verilerin bilgisayar hafızasında tutulduğu fiziki alan adres olarak tanımlanabilir.
- Adres, hem donanımla hem de yazılımla ile ilişkilidir. Donanımsal açıdan adres bellekte yer gösteren bir sayıdan ibarettir.
- Mikroişlemci bellekte bir bölgeye ancak o bölgenin adres bilgisiyle erişebilir.
- Fiziki olarak bilgilerin yerleşimi işlemciden işlemciye göre değişiklik göstermektedir. Verilerin 0000 adresinden artan sırayla yerleştirilmesine doğrusal adresleme denir.
- Bazı işlemciler ise bu yerleşim hafızanın en üst kısmından (FFFF adresinden) aşağı doğru olmaktadır.

## Pointers (İşaretçiler)

- Nesnelerin adresleri, sistemlerin çoğunda, derleyici ve programı yükleyen işletim sistemi tarafından ortaklaşa olarak belirlenir.
- Nesnelerin adresleri program yüklenmeden önce kesin olarak bilinemez ve programcı tarafından da önceden tespit edilemez.
- Programı yazan yada kullanan, nesnelerin adreslerini ancak programın çalışması esnasında (run time) görebilir.

## Pointers (İşaretçiler)

- C++ 'da oluşturduğumuz her tip hafızada belirli byte boyutunda yer kaplar.

```
char    = 1 byte  
int     = 4 byte  
float   = 4 byte  
double  = 8 byte
```

# Pointers (İşaretçiler)

- Pointerlar ise tipe bakmaksızın her zaman;
  - 32 bit sistemlerde 4 byte,
  - 64 bit sistemlerde 8 byte yer kaplar.
- Kısaca; bir integer değişkenin hafızada integer değeri, ya da bir double değişkenin ondalıklı sayı tutması gibi pointer da adres değeri tutan bir değişken olarak tanımlayabiliriz.

# Pointers (İşaretçiler) Bellekte kapladığı alan

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5
6      int i;
7      double d;
8      float f;
9      char c;
10     int *pi;
11     double *pd;
12     float *pf;
13     char *pc;
14
15     cout<<"int i="<<sizeof(i)<<" bayt"<<endl;
16     cout<<"double d="<<sizeof(d)<<" bayt"<<endl;
17     cout<<"float f="<<sizeof(f)<<" bayt"<<endl;
18     cout<<"char c="<<sizeof(c)<<" bayt"<<endl;
19     cout<<endl;
20     cout<<"int pi="<<sizeof(pi)<<" bayt"<<endl;
21     cout<<"double pd="<<sizeof(pd)<<" bayt"<<endl;
22     cout<<"float pf="<<sizeof(pf)<<" bayt"<<endl;
23     cout<<"char pc="<<sizeof(pc)<<" bayt"<<endl;
24
25     return 0;
26 }
```

C:\Users\monster\Desktop\ders

```
int i=4 bayt
double d=8 bayt
float f=4 bayt
char c=1 bayt
```

```
int pi=8 bayt
double pd=8 bayt
float pf=8 bayt
char pc=8 bayt
```

```
-----
Process exited af
Press any key to
```

# Pointers (İşaretçiler)

- Her değişkenin tipi, adı, değeri ve bellekte bulunduğu bir adresi olduğunu biliyoruz. Pointer ise bu yerin yani bellek alanındaki yerin adresinin saklandığı değişken türüdür.

- Örnek Verirsek;

```
okul_no = 453 değişkeni için;  
Tipi      = int  
Adi       = okul_no  
Değeri    = 453  
Adresi    = 1005
```

- 1-A dersliğine ait 1005 adresindeki sıra yeri sabittir. Derse değişik öğrencilerin gelmesi durumunda değişen sadece öğrenci numaralarıdır.

## 1-A DERSLİĞİ

1001, 123	1011, 789	1021, 823
1002, 752	1012, 111	1022, 901
1003, 696	1013, 222	1023, 903
1004, 678	1014, 333	1024, 907
1005, 453	1015, bos	1025, boş
1006, 287	1016, 899	1026, boş
1007, 900	1017, 890	1027, 278
1008, 876	1018, bos	1028, boş
1009, boş	1019, boş	1029, boş
1010, boş	1020, boş	1030, boş

# Referans Operatörü

- Pointerlara, veriler değil de, o verilerin bellekte saklı olduğu bellek gözlerinin başlangıç adresleri atanır. Bir pointer, diğer değişkenler gibi, sayısal bir değişkendir. Bu sebeple kullanılmadan önce program içinde bildirilmelidir.
- Pointerlar konusunda bilmemiz gereken 2 adet önemli operatör vardır. Bunlar;
  - **Reference &** ve (Referans Operatörü)
  - **Dereference \*** operatörleridir. (Referanstan Ayırmak)
- Pointerlar tek başlarına çalışamazlar ve değer alamazlar bunun için başka bir değişkeni referans almak zorundadırlar.



## Referans Operatörü

- İşaretçi, bir değişkenin bellekteki adresini tutan bir değişken gibi düşünülür.
- Örneğin, **a** değişkeninin bellekteki konumunu, yani adresini göz önüne alalım. Bu adresi bir başka **b** değişkeni içine yerleştirelim.
- Bu durumda “**a, b’ nin göstergesidir**” ya da “**a, b’ yi işaret eder**” denir. İşaretçi değişkeninin bildirimi yapılır.
  - Bunun için “**\***” işleci kullanılır.
- İşaretçi değişkene bir adres değişken adresi atanır.
  - Değişkenin adresini bulmak için “**&**” işleci kullanılır.

# Pointer Tipindeki Değişkenlerin Tanımlanması

Bir pointer aşağıdaki gibi tanımlanır;

- **Tip** \*degisken\_adi;
- Değişken Türü: int, double, char, string olabilir.
- \* işaretinin iki kullanım amacı vardır.
  - Eğer tanımlama aşamasında değişkenin önüne getirilirse o değişkenin pointer olduğu belirtilir.
  - Eğer kod içerisinde bir işaretçi değişkenin önüne getirilirse o değişkende kayıtlı adres üzerindeki değeri gösterir.

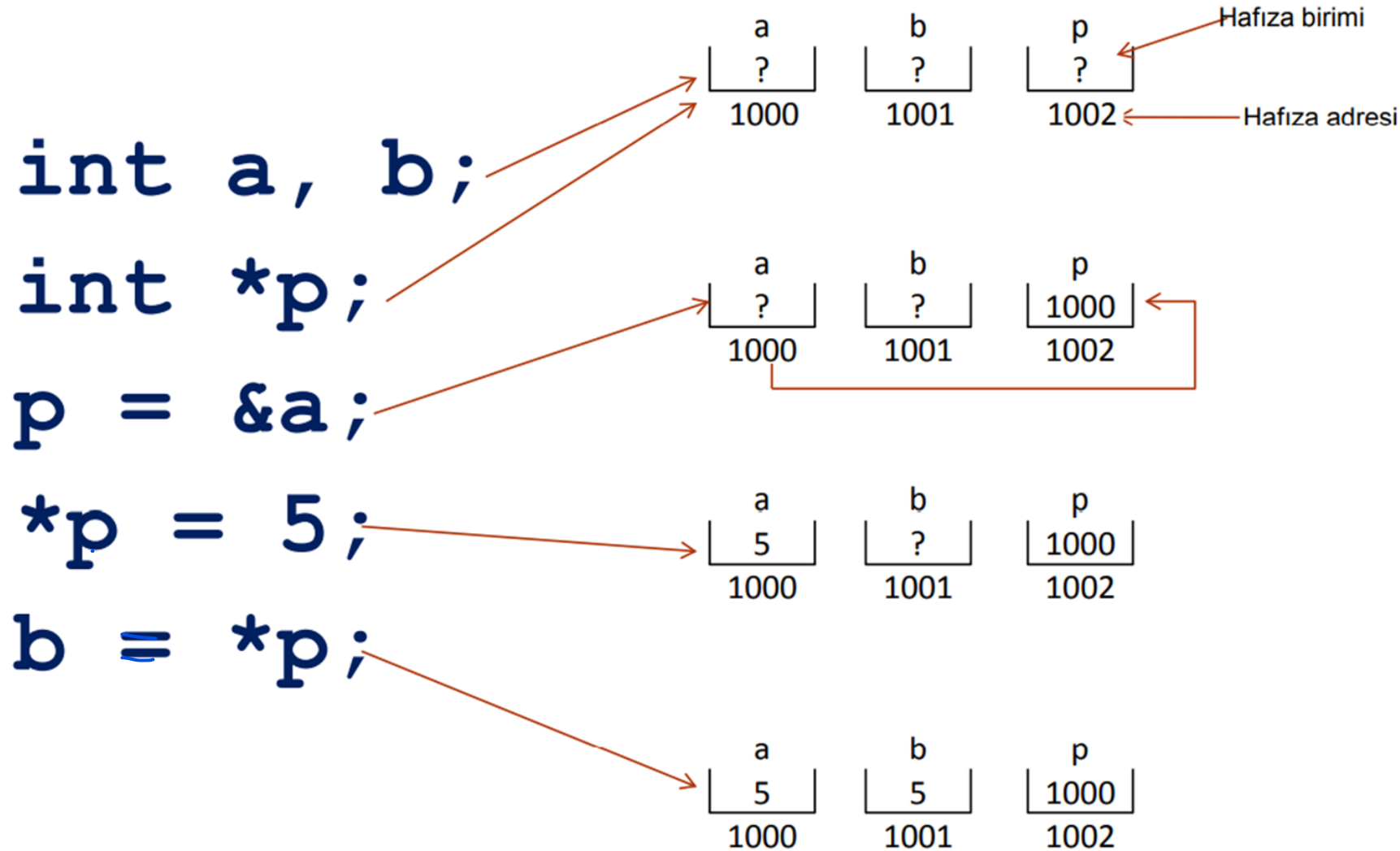
# Pointer Tipindeki Değişkenlerin Tanımlanması

Referans operatörü (&) önüne geldiği değişkenin hafızada saklandığı yeri yani adresini gösterir.

```
int *b;  
int okul_no=453;  
b=&okul_no;           //1005 !!!dikkat b atanırken * yok.
```

- Burada b isimli işaretçi okul\_no isimli değişkeni referans olarak almaktadır ve “b eşittir okul\_no’nun adresi” diyebilmekteyiz.
  - Diğer bir deyişle, b değişkeni integer tipindeki herhangi bir değişkenin adresini tutabilecek olan bir işaretçidir.
- Ayrıca b değişkenine okul\_no nun bellek adresini atayabilmemiz için b yi int \*b şeklinde yani integer pointer olarak tanımlıyoruz.
- **b=&okul\_no** işleminde b’ye okul\_no değişkenin referans değerinin (yani adresi) atanması için \* işareti koyulmamıştır. Bu diğer değişkenlerde (int, char v.b.) yapılan işlemin bir benzeridir.

# Pointer Tipindeki Değişkenlerin Tanımlanması



# Pointer Tipindeki Değişkenlerin Tanımlanması

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5      int a=25;
6      int b=a;
7      int *c=&a;
8
9      cout<<"a="<<a<<endl;
10     cout<<"b="<<b<<endl;
11     cout<<"c="<<c<<endl;
12     cout<<endl<<endl;
13     cout<<"&a="<<&a<<endl;
14     cout<<"&b="<<&b<<endl;
15     cout<<"&c="<<&c<<endl;
16
17
18     return 0;
19 }
```

C:\Users\mon...

a=25  
b=25  
c=0x6ffe0c

&a=0x6ffe0c  
&b=0x6ffe08  
&c=0x6ffe00

-----  
Process exited  
Press any key

# Pointer Tipindeki Değişkenlerin Tanımlanması

```
int *b;  
int okul_no=453;  
b=&okul_no;  
cout << b //1005  
cout << *b //453
```

Bu durumda;

```
cout << *okul_no << endl; //HATA  
cout << *&okul_no << endl; //453  
cout << &*okul_no << endl; //HATA  
cout << &okul_no << endl; //1005
```

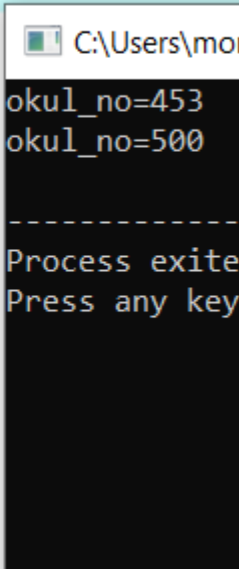
Atanmamış adrese  
çağrı yapılıyor.

Adrese atama  
yapılıp tekrar  
çağrılıyor.

Atanan adres  
çağrılıyor.

# Pointer Tipindeki Değişkenlerin Tanımlanması

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5      int okul_no;
6      int *b;
7      okul_no=453;
8      cout<<"okul_no="<<okul_no<<endl;
9      b=&okul_no;
10     *b=500;
11     cout<<"okul_no="<<okul_no<<endl;
12
13     return 0;
14 }
```



- burada 453 numaralı öğrencinin oturduğu sıranın adresi b değişkenine & referans operatörüyle aktarılıyor.
- b değişkeni ne 453 nolu öğrencinin oturduğu adres atanıyor.
- Daha sonra dereference (\*b=500) ile 453 nolu öğrenci bu sıradan kaldırılarak yerine 500 nolu öğrenci oturtuluyor.
- Artık okul\_no=500 oldu.



# Pointer Aritmetiği

- Pointer üzerinde yapılabilecek iki tip aritmetik işlem vardır:
  - 1. Toplama
  - 2. Çıkarma.
- Ancak eklenecek yada çıkartılacak değer tamsayı olmalıdır
- Pointer değişkenin değeri 1 arttırıldığı zaman değişken bir sonraki veri bloğunu işaret eder. Değişkenin alacağı yeni değer pointer değişkenin ne tip bir veri bloğunu işaret ettiğine bağlıdır.
- Bir pointer her arttırıldığında, hafızada aynı tipteki bir sonraki değeri, azaltıldığında ise hafızada aynı tipteki bir önceki değeri gösterir.



# Pointer Aritmetiği

```
1  #include <iostream>
2  using namespace std;
3  int main(int argc, char** argv) {
4      int *p1, i=3;
5      p1=&i;
6
7      double *p2, j=5;
8      p2=&j;
9
10     cout<<"i nin boyutu="<<sizeof(i)<<" byte"<<endl;
11     cout<<"p1 nin boyutu="<<sizeof(p1)<<" byte"<<endl;
12     cout<<"p1="<<p1<<endl;
13     p1++;
14     cout<<"p1="<<p1<<endl<<endl<<endl;
15
16     cout<<"j nin boyutu="<<sizeof(j)<<" byte"<<endl;
17     cout<<"p2 nin boyutu="<<sizeof(p2)<<" byte"<<endl;
18     cout<<"p2="<<p2<<endl;
19     p2++;
20     cout<<"p2="<<p2<<endl;
21
22     return 0;
23 }
```

C:\Users\monster\Desktop

```
i nin boyutu=4 byte
p1 nin boyutu=8 byte
p1=0x6ffe0c
p1=0x6ffe10
```

```
j nin boyutu=8 byte
p2 nin boyutu=8 byte
p2=0x6ffe00
p2=0x6ffe08
```

```
-----
Process exited after 0
Press any key to conti
```

HEX	6F FE0C	HEX	6F FE10
DEC	7.339.532	DEC	7.339.536

HEX	6F FE00	HEX	6F FE08
DEC	7.339.520	DEC	7.339.528

# Pointer ve Diziler

- C++' da diziler ve pointerlar iç içedir ve çoğu yerde birbirlerinin yerinede kullanılabilirler. Dizi ismini sabit bir pointer olarak düşünebiliriz. Pointerlar dizilerin bir çok işleminde kullanılabilir.
- `int *p, x[5];`
- `p=x;`
  - Veya
- `p=&x[0];`

# Pointer ve Diziler

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5      int i[5]={1,2,3,4,5};
6      int *ptr;
7      ptr=i;
8      cout<<"i="<<&i<<endl;
9      cout<<"ptr="<<ptr<<endl;
10     cout<<"ptr adresindeki deger="<<*ptr<<endl;
11     return 0;
12 }
```

C:\Users\monster\Desktop\d

```
i=0x6ffdf0
ptr=0x6ffdf0
ptr adresindeki deger=1
```

```
-----
Process exited after 0.05
Press any key to continue
```

# Pointer ve Diziler

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5      int i[5]={1,2,3,4,5};
6      int *ptr;
7      ptr=i;
8      cout<<"i="<<&i<<endl;
9      cout<<"ptr="<<ptr<<endl;
10     cout<<"ptr adresindeki deger="<<*ptr<<endl;
11     ptr++;
12     cout<<"ptr="<<ptr<<endl;
13     cout<<"ptr adresindeki deger="<<*ptr<<endl;
14     ptr++;
15     cout<<"ptr="<<ptr<<endl;
16     cout<<"ptr adresindeki deger="<<*ptr<<endl;
17
18     return 0;
19 }
```

```
C:\Users\monster\Desktop\d
i=0x6ffdf0
ptr=0x6ffdf0
ptr adresindeki deger=1
ptr=0x6ffdf4
ptr adresindeki deger=2
ptr=0x6ffdf8
ptr adresindeki deger=3
-----
Process exited after 0.13
Press any key to continue
```

# Pointer ve Diziler

```
1  #include <iostream>
2  using namespace std;
3
4  int main(int argc, char** argv) {
5      int i[5]={1,2,3,4,5};
6      int *ptr;
7      ptr=i;
8      cout<<"i="<<&i<<endl;
9      cout<<"ptr="<<ptr<<endl;
10     cout<<"*ptr adresindeki deger="<<*ptr<<endl;
11     cout<<"(ptr+2) un adresi="<<(ptr+2)<<endl;
12     cout<<"*(ptr+2) adresindeki deger="<<*(ptr+2)<<endl;
13     cout<<"ptr[2] adresindeki deger="<<ptr[2]<<endl;
14
15
16     return 0;
17 }
```

C:\Users\monster\Desktop\ders sl

```
i=0x6ffdf0
ptr=0x6ffdf0
*ptr adresindeki deger=1
(ptr+2) un adresi=0x6ffdf8
*(ptr+2) adresindeki deger=3
ptr[2] adresindeki deger=3

-----
Process exited after 0.1371 s
Press any key to continue . .
```