SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

İŞARETÇİLER
(Pointers)

Prof. Dr. Cemil ÖZ
Prof. Dr. Celal ÇEKEN
Doç. Dr. Cüneyt BAYILMIŞ

Konular

- ✓ İşaretçiler (Pointers)
- ✓ İşaretçiler ve Diziler
- ✓ İşaretçi Aritmetiği
- ✓ İşaretçi Gösteren İşaretçiler
- ✓ Struct Gösteren İşaretçiler
- ✓ Fonksiyon Parametresi Olarak İşaretçiler
- ✓ Dinamik Bellek Kullanımı (Nesne Gösteren İşaretçiler)
- ✓ Dinamik Diziler
- ✓ Void İşaretçi
- ✓ Örnek
- ✓ Kaynaklar

İşaretçiler (Pointers)

- ✓ **Çok güçlü bir özellik fakat yönetimi zor.**
- ✓ **Belleğin dinamik olarak kullanımına olanak sağlar.**
- ✓ **Özellikle dizi ve karakter dizileriyle yakın ilişkisi vardır.**

Genel kullanım alanları

- ✓ Dizi elemanlarına erişim
- ✓ Fonksiyon giriş parametrelerinin orijinalinde de değişiklik gerektiğinde
- ✓ Dizi ve karakter dizilerini fonksiyonlara gönderirken
- ✓ Sistemden bellek isterken
- ✓ Bağlı listeler, ağaç yapıları v.s. gibi veri yapıları oluştururken

İşaretçiler (Pointers)

İşaretçi değişkenler

✓ Değer olarak bellek adresi içerir.

✓İşaretçinin içerdiği adres, bir değişken, fonksiyon, yapı veya nesne gösterebilir.

✓ Normalde değişkenler belirli bir değeri içerir. (direct reference)

✓ İşaretçiler belirli değere sahip olan değişkenin bellek adresini içerir.
(indirect reference)

İşaretçiler (Pointers)

İşaretçi işleçleri

& adres işleci, değişkenin bellek adresini döndürür.

* adres içeriği ilgili adresteki değeri (veriyi) döndürür.

& ve * birbirinin tümleyenidir (tersidir).

Örnek:

```
int a=5;
char b='z';
char *ptr_char=NULL;
ptr_char=&b;
```

Adres	Değişken	Değer
0100	a	5
0101		
0102	b	'z'
0103	ptr_char	0102
0104		
0105		
0106		
0107		

İşaretçiler (Pointers)

İşaretçi değişkenlerin bildirimi

***tip * isaretci_adi*;**

int *myPtr;
int *myPtr1, *myPtr2;

Alacağı değerler : **0**, **NULL**, ya da herhangi bir adres
0 or **NULL** hiçbir şeyi göstermez.

İşaretçiler (Pointers)

- ✓ **&** (adres işleci) ***"address of"***

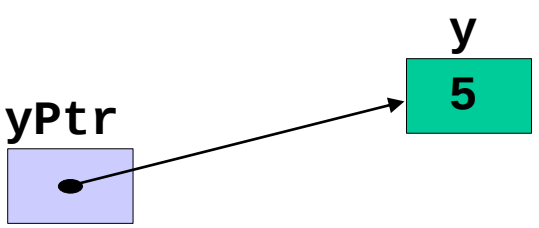
Bellek adresini geri döndürür.

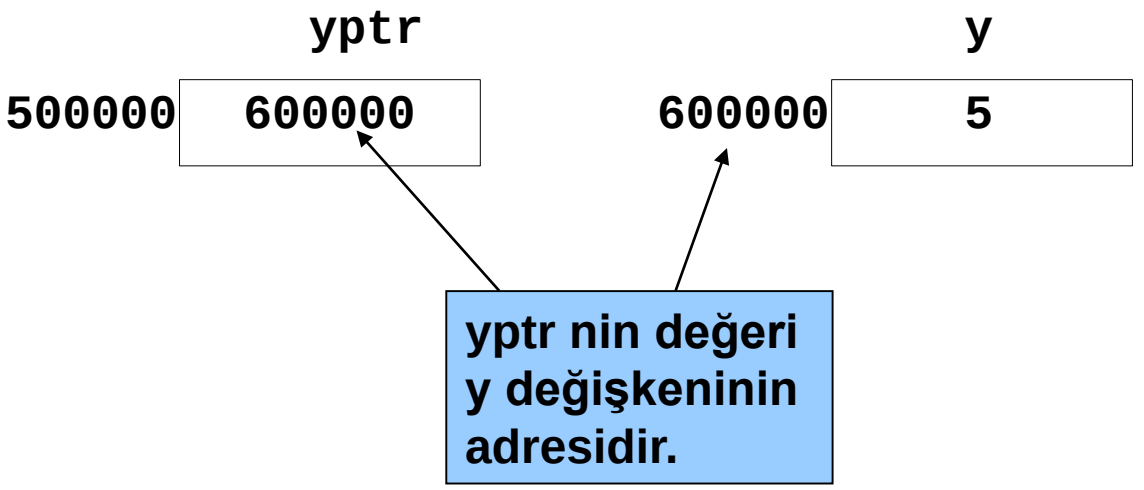
İlk değer ataması:

int sayi;
int *p1 =&sayi;

```
int y = 5;
int *yPtr;
yPtr = &y;    // yPtr gets address of y
```

- ✓ **yPtr** "points to" **y**





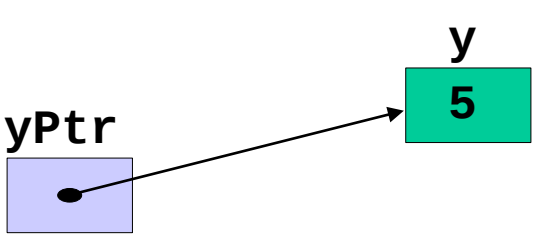
İşaretçiler (Pointers)

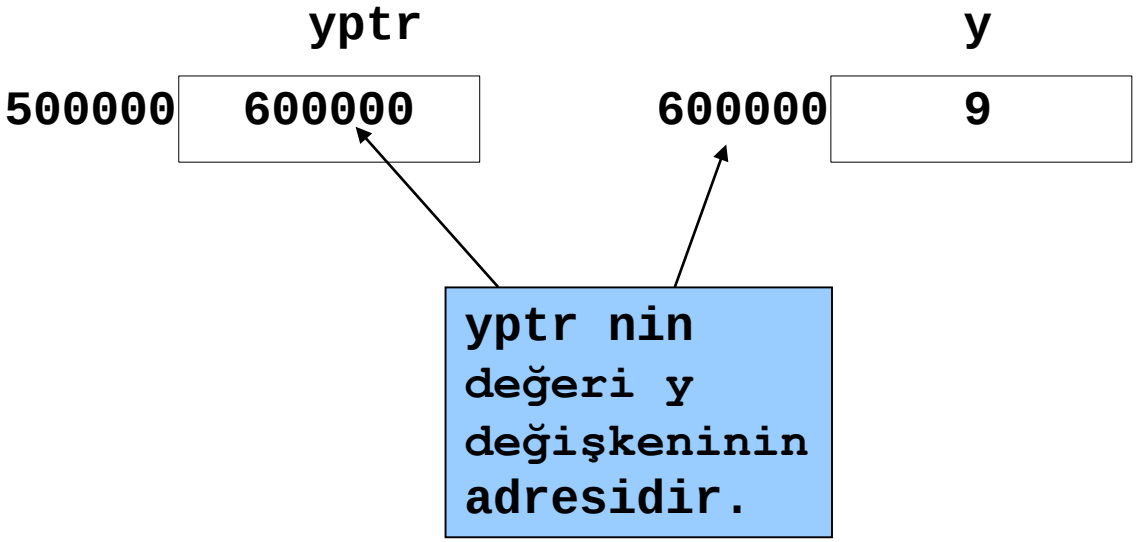
- ✓ ***** (adres içeriği) ***"value pointed by"***

Bellek adresini geri döndürür.

```
int y;
int *yPtr;
yPtr = &y;
*yptr=9;
```

- ✓ ***** ve **&** birbirlerinin tersidir.





İşaretçiler (Pointers)

```
#include <iostream>
using namespace std;
int main(){
        int x;
        int *xptr;
        x=3;
        xptr=&x;
        cout<<"x'in adresi: "<< &x <<endl;
        cout<<"x'in icerigi"<< xptr <<endl;
        cout<<"x'in degeri"<< x <<endl;
        cout<<"*xptr'nin degeri"<< *xptr <<endl;
        cout<<"* ve & operatorleri birbirinin tümleyenidir \n";
        cout<<"&*xptr: "<< &*xptr <<endl;
        cout<<"*&xptr: "<< *&xptr <<endl;
        return 0;
}
```

İşaretçi Aritmetiği

✓ İşaretçilerle kullanılabilecek aritmetik işleçler:

++, --, +, +=, -, -=

Örnek:

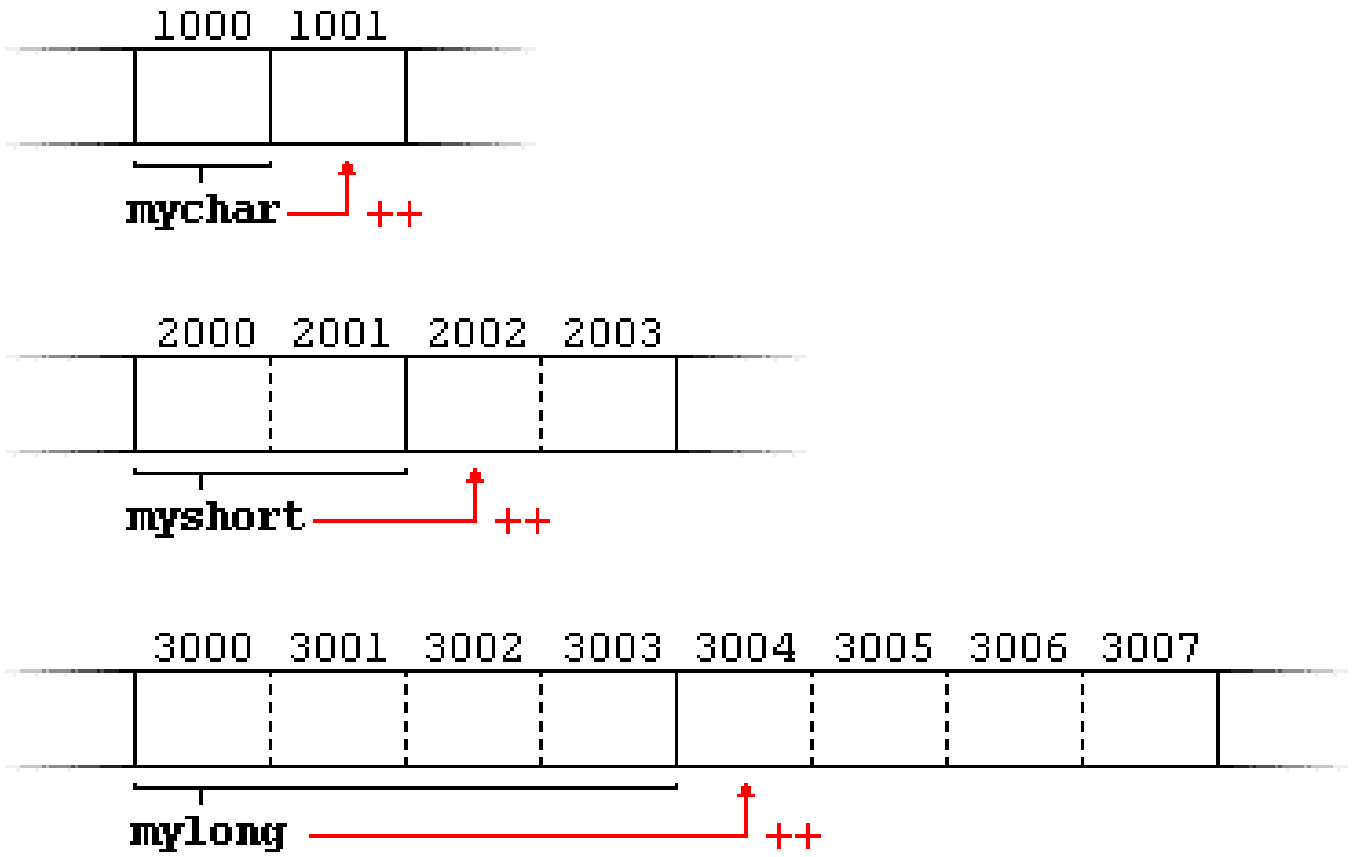
```
int a[5]={0};
    // a dizisinin  başlangıç adresi 0100 olsun.

int *aPtr=a;
    //aPtr, a dizisinin başlangıcını işaret ediyor
    // yani, aPtr=0100 dür.

aPtr++;
    // aPtr=0102 olur. a[1] elamanını işaret ediyor

(*aPtr)++;
    //a[1]=1 olur.
```

Adres	İçerik
0100	0
0101	
0102	0 ->1
0103	
0104	0
0105	
0106	0
0107	
0108	0
0109	
010A	

İşaretçi Aritmetiği

```
char *mychar;
short *myshort;
long *mylong;
```

```
mychar++;
myshort++;
mylong++;
```



1isaretciAritmetik.cpp

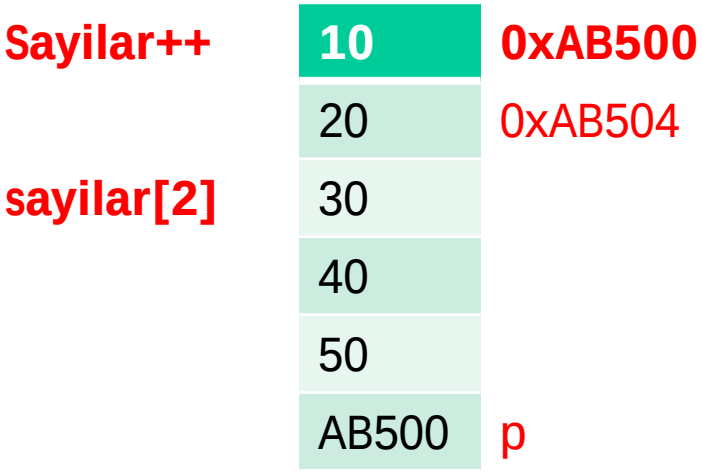
İşaretçiler ve Diziler

- ✓ **Dizinin ilk elemanının adresini içeren dizi adı aslında işaretçi gibi düşünülebilir.**
- ✓ **İşaretçiler dizi elemanlarının işlemlerinde kullanılabilir.**

```
int    sayilar[20];
int *p;
p=sayilar;
P=&sayilar[0];
```

Örnek: sayılar dizisinin 3 numaralı elemanına 5 değeri atama
sayilar [3] = 5 ya da *(p+3)=5

```
#include <iostream.h>
int main ()
{
  int sayilar[5];
  int * p;
  p = sayilar;  *p = 10;
  p++;  *p = 20;
  p = & sayilar[2];  *p = 30;
  p = sayilar + 3;  *p = 40;
  p = sayilar;  *(p+4) = 50;
  for (int n=0; n<5; n++)
    cout << sayilar[n] << ", ";
  return 0;
}
```



İşaretçiler ve Diziler

✓ **Dizinin ilk elemanının adresini içeren dizi adı aslında işaretçi gibi düşünülebilir.**

```
int sayilar[20];
int *p;
p=sayilar
```

```
int sayi;
int *p1 =&sayi;
```

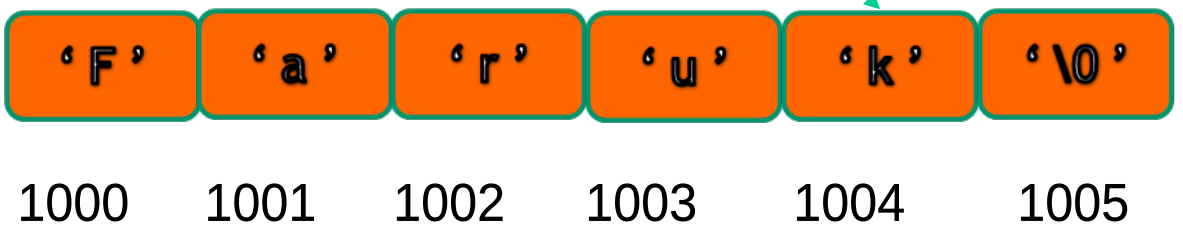
```
#include <iostream.h>
int main ()
{
  int sayilar[5];
  int * p;
  p = sayilar;  *p = 10;
  p++;  *p = 20;
  p = & sayilar[2];  *p = 30;
  p = sayilar + 3;  *p = 40;
  p = sayilar;  *(p+4) = 50;
  for (int n=0; n<5; n++)
    cout << sayilar[n] << ", ";
  return 0;
}
```

```
int a[5] = {0,1,2,5,7};
cout<<*(a+3);
cout<<*(a++);  //hatalı, a işaretçi sabitidir,
                // değeri değiştirilemez


int main()
{
   int intarray[5] = { 31, 54, 77, 52, 93 };
   for(int j=0; j<5; j++)
      cout << *(intarray+j) << endl;
   return 0;
 }
```

Karakter Katarları

char *katar= “Faruk";

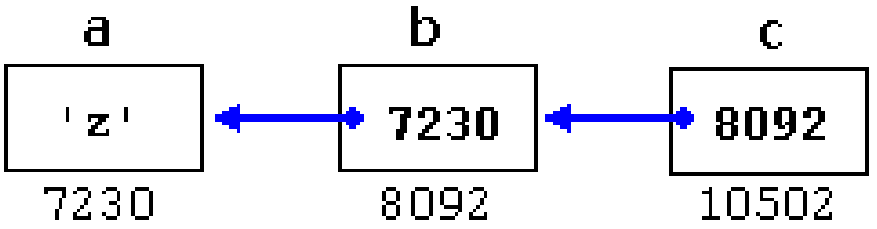
*(katar+4)=‘k’
katar[4]= ‘k’



katar=1000

İşaretçi Gösteren İşaretçiler

```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```



Struct Gösteren İşaretçiler

```
#include <iostream>
using namespace std;

struct Olcu
{
   int metre;
   int cmetre;
};

int main()
{
   Olcu *d1=new Olcu(); //Belleğin heap bölgesinden yer açar

   //d1 = (Olcu *)malloc(2*sizeof(Olcu)); //C tarzı yer ayırma
   cout << "\nuzunluk(metre) giriniz: ";
   cin >> d1->metre;
   cout << "uzunluk(cmetre) giriniz:: ";
   cin >> d1->cmetre;

   cout << d1->metre << "m," << d1->cmetre << "cm + ";

   system("pause");
   return 0;
}
```

new komutuyla ayrılan yerlerin delete ile boşaltılması gereklidir. Aksi taktirde bellek sızıntıları (memory leak) meydana gelir. Özellikle uzun süreli çalışacak ya da fazla miktarda dinamik bellek kullanan programlarda bellek sızıntıları çok büyük sorunlara neden olabilir. Program sonlandığında ayrılan yerlerin kontrolü işletim sistemine geçer.

2Struct.cpp

Fonksiyon Parametresi Olarak İşaretçiler

- ✓ Bir fonksiyona parametre aktarmanın iki yolu vardır.
  - ✓ Değerle Çağırma
  - ✓ Adresle Çağırma
    - ✓ Adresle Çağırma İşaretçiler ile yapılır.

1. Yöntem, değerle çağırma

```
#include <iostream>
using namespace std;
void arttir (int x) {
    x++;
}
int main(){
    int sayi=7;
    cout<<"sayi: "<< sayi <<endl;
    arttir (sayi);
    cout<<"sayi: "<< sayi <<endl;
    return 0;
}
```

2. Yöntem, adresle çağırma

```
#include <iostream>
using namespace std;
void arttir (int *x) {
    (*x)++;
}
int main(){
    int sayi=7;
    cout<<"sayi: "<< sayi <<endl;
    arttir (&sayi);
    cout<<"sayi: "<< sayi <<endl;
    return 0;
}
```

Fonksiyon Parametresi Olarak İşaretçiler

```
void centimize(double*);

int main()
 {
   double var = 10.0;
   cout <<“var = ”<<var<<“ inches”;
   centimize(&var);
   cout <<“var :”<<var<<“cmeters”;
   return 0;
 }
//-----------------------------------------------
void centimize(double* ptrd)
{
   *ptrd *= 2.54;
}
```

```
#include <iostream>
#include <iomanip>
#include <math.h>

using namespace std;


void centimize(double*);  //bildirim

#define MAX 5

int main()
{
   double varray[MAX]={ 10.0, 43.1, 95.9, 59.7, 87.3 };
   centimize(varray);
   for(int j=0; j<MAX; j++)
      cout << "vararray[" << j <<"]="
           << varray[j] << " centimeters" << endl;
   system("pause");
   return 0;
}


void centimize(double* ptrd)
{
   for(int j=0; j<MAX; j++)
      *ptrd++ *= 2.54;  //ptrd varray dzs elemanlarına işaret eder
}
```

FonksiyonParametresi.cpp

Fonksiyon Parametresi Olarak İşaretçiler (const)

Gönderilen adresin içeriğinin değiştirilememesini garanti etmek için const ifadesi kullanılabilir.

```
void copystr(char*, const char*);  //prototype

int main()
{

   char* str1 = "Self-conquest is the greatest victory.";
   char str2[80];              //empty string

   copystr(str2, str1);        //copy str1 to str2
   cout << str2 << endl;       //display str2
   return 0;
}
//-------------------------------------------------------------
void copystr(char* dest, const char* src)
{
   while( *src )               //until null character,
      *dest++ = *src++;        //copy chars from src to dest
   *dest = '\0';               //terminate dest
}
```

In the declaration of strcpy() just shown, the argument const char* src specifies that the characters pointed to by src cannot be changed by strcpy(). It does not imply that the src pointer itself cannot be modified. To do that the argument declaration would need to be char* const src.

Fonksiyon Parametresi Olarak İşaretçiler

```
#include <iostream>
#include <cctype> // islower ve toupper

using namespace std;

void buyukHarfeCevir( char * );

int main()
{
      char metin[] = "Elektronik ve Bilgisayar Egitimi";


      cout << "Cevirme isleminden onceki metin:" << metin;
      buyukHarfeCevir( metin );
      cout << "\nCevirme isleminden sonraki metin:" << metin<< endl;

      system("pause");
      return 0;

} //  main

void buyukHarfeCevir( char *sPtr )
{
      while ( *sPtr != '\0' ) // metini son karaktere kadar tara
      {
         if ( islower( *sPtr ) ) // Karakter kucuk harf ise,
            *sPtr = toupper( *sPtr ); // buyuk harfe cevir

         sPtr++;
      } //  while
} // buyukHarfeCevir
```

Soru: Önceki derslerde kullanılan şifreleme programını fonksiyon kullanarak yapınız.

Şifreleme işlemi için : void sifrele (char* mesaj)

Şifre çözme işlemi için: void sifreCoz (char* mesaj)

Fonksiyonları kullanılmalıdır…

BuyukHarfeCevir.cpp

Fonksiyon İşaretçileri

- ✓ Bir fonksiyona işaret ederler
- ✓ Fonksiyon işaretçisi, bir fonksiyona parametre olabilir
- ✓ Fonksiyon işaretçisi bir fonksiyondan geri döndürülebilir
- ✓ Birden fazlası dizi olarak tanımlanabilir
- ✓ Başka bir fonksiyon işaretçisine atanabilir

- ✓ Tanımlama:
- ✓ int (*fonk1) (int); // int parametresi alan ve int döndüren bir fonksiyon işaretçisi
- ✓ int* fonk2(int,int); // iki adet int parametre alan ve int işaretçisi döndüren fonksiyon

Fonksiyon İşaretçileri

```
#include <iostream>
using namespace std;
int kareAl(int x)
{
    return x*x;
}
int kubAl(int x)
{
     return x*x*x;
}
```

```
int main(){
      int (*islem)(int);
      int i;
      char c;
      cout<<" karesi-1,  kübü-2 ? ";
      cin>>c;
      cout<<"\n Sayıyı gir : ";
      cin >> i;
      if (c == '1')
            islem = kareAl;
      else if(c=='2')
            islem = kubAl;
      else{
            cout<<"Yanlis secim";
            return 0;
      }
      cout<<"Sonuc = "<< islem(i) << endl;
}
```

kareAl fonksiyonunun başlangıç adresi islem fonksiyon pointerına atanıyor

Seçilen fonksiyon çalıştırılıyor

Dinamik Bellek Yönetimi

- ✓ new ve delete operatörleri
- ✓ new veri tipi [uzunluk]
- ✓ new komutu ile tahsis edilen yerin başlangıç adresi döndürülür
- ✓ Örnek:

```
int *ptr;

p= new int [3];      // 3*sizeof(int) kadar yer tahsis et

delete ptr;          // ptr için tahsis edilen yeri serbest bırakır (boşaltır)

delete [] ptr;       // elemanlar dizisi için tahsisli yeri boşaltır
```

Dinamik Bellek Tahsisinin Kontrolü

✓ Bellek tahsisinin başarılı olup olmadığı C++'da 2 yol ile belirlenir.

❶ Exception (istisna) kullanımı

- Yer tahsisi başarısız olduğunda bad_alloc tipinde bir istisna yollanır.
- bad_alloc istisnası yollandığında programın çalışması sonra erdirilir.

❷ Nothrow

- Program sonlandırılmaz ve istisna yollanmaz.
- new operatörü tarafından boş (null) bir pointer döndürülür.
- Program çalışmayı devam ettirir.

Örnek:

```
int *ptr;
ptr= new (nothrow) int [3];
if (ptr)
        cout<<"Bellek Tahsisi Gerçekleştirildi";
else
        cout<<"Bellek Tahsisi Gerçekleştirilmedi";
```

Dinamik Diziler

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main ()
{
  char input [100];
  int i,n;
  long *l, total = 0;
  cout << "Kac sayi gireceksiniz?";
  cin>>n;
  l= new long[n];
  if (l == NULL) exit(1);
  for (i=0; i<n; i++)
  {
    cout << "sayiyi giriniz ";
    cin>>l[i];
  }
  cout << "Girdiğiniz degerler: ";
  for (i=0; i<n; i++)
     cout << l[i] << ", ";
  delete[] l;
  system("pause");
  return 0;
}
```

- ✓ program çalışırken dinamik olarak bellek ayırmak(new)/silmek(delete).
- ✓ bellek ayrılamaz ise NULL değeri döndürülür.

DinamikDiziler.cpp

Dinamik Diziler

```
int main()
{
    int elemanSayisi=0;
    cin>>elemanSayisi;

    int* ptrInt=new int[elemanSayisi]; //heap bölgesinde elemanSayisi tamsayılık yer açıldı...

    cout <<ptrInt; // (0x8fcd008) heap bolgesinde acilan
    //10 int lik yerin başlangıç adresi. ptrInt yığın bölgesinde saklanır.

    ptrInt[0]=10;
    cout<<*(ptrInt+0); //10 değerini yazar...

    char* mesaj= new char[10];

    // mesaj işaretçisinin gösterdiği adresi yazdırınız

    delete [] ptrInt;
    delete [] mesaj;

    ptrInt=NULL;
    mesaj=NULL;

    return 0;
}
```

DinamikDiziler.cpp

Dinamik Diziler

```
struct Ogrenci
{
        char numara[10];
        char ad[10];
        char  soyad[10];
        float genelOrtalama;
};

Ogrenci *ogrenciler[100];

int ogrenciSayisi = 0;

cin >> ogrenciSayisi;

for(int i=0;i<ogrenciSayisi; i++)
{
     ogrenciler[i]=new Ogrenci();
}

for (int i = 0; i < ogrenciSayisi; i++)
{
    cin>>ogrenciler[i]->numara;
    cin>> ogrenciler[i]->ad;
    cin>> ogrenciler[i]->soyad;
    //getline(cin,ogrenciler[i].genelOrtalama);
    cin>>ogrenciler[i]->genelOrtalama;// = 8;
    cout << "\n----------------" << endl;
}
```

- ✓ program çalışırken dinamik olarak bellek ayırmak(new)/silmek(delete).
- ✓ bellek ayrılamaz ise NULL değeri döndürülür.

DinamikDiziler2.cpp

Dinamik Diziler

```
struct Ogrenci
{
        char numara[10];
        char ad[10];
        char  soyad[10];
        float genelOrtalama;
};
```

- ✓ program çalışırken dinamik olarak bellek ayırmak(new)/silmek(delete).
- ✓ bellek ayrılamaz ise NULL değeri döndürülür.

```
cin >> ogrenciSayisi;

Ogrenci* ogrenciler= new Ogrenci[ogrenciSayisi];

for(int i = 0; i < ogrenciSayisi; i++)
{
    cin>>ogrenciler[i].numara;
    cin>> ogrenciler[i].ad;
    cin>> ogrenciler[i].soyad;
    //getline(cin,ogrenciler[i].genelOrtalama);
    cin>>ogrenciler[i].genelOrtalama;// = 8;
    cout << "\n-----------------" << endl;
}
```

DinamikDiziler3.cpp

Dinamik Diziler

```
cout<<"kac ogrenci kaydetmek istiyorsunuz";
cin>>boyut;
Ogr *pogr= new Ogr [boyut];

for(int i=0;i<boyut;i++)
{
    cout<<"Ad giriniz: ";                cin>> pogr->ad;cout<<endl;
    cout<<"Soyad giriniz: ";             cin>> pogr->soyad;cout<<endl;
    cout<<"Ogrenci No giriniz: ";        cin>> pogr->numara;cout<<endl;
    cout<<"Ortalama giriniz: ";          cin>> pogr->ortalama;cout<<endl;
    pogr++;
}
pogr-=boyut;
cout<<setw(20)<<"AD"<<setw(20)<< "SOYAD"  <<setw(20)<<   "NUMARA"  <<setw(

for(int i=0;i<boyut;i++)
{
    cout<<setw(20)<< pogr->ad;
    cout<<setw(20)<< pogr->soyad;
    cout<<setw(20)<< pogr->numara;
    cout<<setw(20)<< pogr->ortalama;
    pogr++;
    cout<<endl;
}
```

DinamikBellekKullanimiIsaretci.cpp **DinamikBellekKullanimiIndexli.cpp**

Dinamik Nesneler 1

DinamikNesneler.cpp | Ogrenci.cpp | Ogrenci.h

```
 * Ogrenci.h

#ifndef OGRENCI_H_
#define OGRENCI_H_

#include <iostream>
#include <string>

using namespace std;
class Ogrenci
{
    private:
        string ad;
        string soyad;
        string numara;
        int notOrtalamasi;
    public:
        Ogrenci();
        virtual ~Ogrenci();
        void bilgiGirisi(); //
        void bilgiYazdir();
        string getAd() const;
        void setAd(string ad);
        int getNotOrtalamasi() const;
        void setNotOrtalamasi(int notOrtalamasi);
        string getNumara() const;
        void setNumara(string numara);
        string getSoyad() const;
        void setSoyad(string soyad);
};
#endif /* OGRENCI_H_ */
```

Ogrenci.h

Dinamik Nesneler 2

DinamikNesneler.cpp | Ogrenci.cpp | Ogrenci.h

```
Ogrenci::Ogrenci()
{
    // TODO Auto-generated constructor stub
    this->setAd("");
    this->setSoyad("");
    this->setNumara("");
    this->setNotOrtalamasi(0);
}

void Ogrenci::bilgiGirisi()
{
    cout<<"Ad giriniz: ";           cin>>ad; cout<<endl;
    cout<<"Soyad giriniz: ";        cin>>soyad;cout<<endl;
    cout<<"Ogrenci No giriniz: ";   cin>> numara;cout<<endl;
    cout<<"Ortalama giriniz: ";     cin>> notOrtalamasi;cout

}

void Ogrenci::bilgiYazdir()
{
    cout<<setw(20)<< ad;
    cout<<setw(20)<< soyad;
    cout<<setw(20)<< numara;
    cout<<setw(20)<< notOrtalamasi;
    cout<<endl<<setw(20)<<"-----------"<<endl;

}

string Ogrenci::getAd() const
{
    return ad;
```

Ogrenci.cpp

Dinamik Nesneler 3

DinamikNesneler.cpp | Ogrenci.cpp | Ogrenci.h

```
// Name        : DinamikNesneler.cpp

#include <iostream>
#include "Ogrenci.h"

using namespace std;


int main()
{
    int ogrenciSayisi=0;

    cin>>ogrenciSayisi;

    Ogrenci* ogrenciler= new Ogrenci[ogrenciSayisi];
    //Ogrenci ogr1("Mine", "Yağız");

    for(int i=0;i<ogrenciSayisi;i++)
        ogrenciler[i].bilgiGirisi();

    for(int i=0;i<ogrenciSayisi;i++)
            ogrenciler[i].bilgiYazdir();

    return 0;
}
```

OgrenciBilgiSistemi.cpp

Dinamik Nesneler 4

```
#include <iostream>
#include "Ogrenci.h"

using namespace std;


int main()
{
    int ogrenciSayisi=0;

    //cin>>ogrenciSayisi;

    //Ogrenci* ogrenciler= new Ogrenci[ogrenciSayisi];

    Ogrenci* ogrenci1=new Ogrenci();
    ogrenci1->bilgiGirisi();
    ogrenci1->bilgiYazdir();

//  for(int i=0;i<ogrenciSayisi;i++)
//      ogrenciler[i].bilgiGirisi();
//
//  for(int i=0;i<ogrenciSayisi;i++)
//          ogrenciler[i].bilgiYazdir();

    return 0;
}
```

OgrenciBilgiSistemi.cpp

Dinamik Nesneler (Örnek Çalışma 1)

```
int main()
{
   Bolum bilgisayarMuhendisligi("Bilgisayar Mühendisliði");
   Bolum emk("Endüstri Müh.");
   Personel* ayse = new Personel("Ayse Aksaç", 45000);
   bilgisayarMuhendisligi.setBolumBaskani(ayse);
   Personel* ahmet = new Personel("Ahmet Mekin", 50000);
   emk.setBolumBaskani(ahmet);
   emk.setBolumKoordinatoru(ayse);
   ahmet->set_maas(55000);
   bilgisayarMuhendisligi.yazdir();
   emk.yazdir();
   delete ahmet;
   delete ayse;

   system("pause");
   return 0;
}
```

Bolum/Bolum.cpp

Dinamik Nesneler (Örnek Çalışma 2)

```
#include "Ogrenci.h"
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
int main()
{
    int ogrenciSayisi=0;

    cin>>ogrenciSayisi;

    Ogrenci* ogrenciler= new Ogrenci[ogrenciSayisi];
    Ogrenci ogr1("Mine", "Yaðýz");

    for(int i=0;i<ogrenciSayisi;i++)
        ogrenciler[i].bilgiGirisi();

    cout<<ogr1.getAd();
    cout<<ogrenciler[0].getAd();
    cout<<(ogrenciler+1)->getAd();

    for(int i=0;i<ogrenciSayisi;i++)
            ogrenciler[i].bilgiYazdir();

    system("pause");
    return 0;
}
```

OgrenciBilgiSistemi.cpp

Dinamik Bellek Kullanımı (Nesne Gösteren İşaretçiler)

Struct gösteren işaretçiler için de benzer işlemler yapılmalıdır…

```
int main()
{
	Personel *p1=new Personel();
	Personel *p2=new Personel();
	//float mesaiUcreti;
	//int mesaiSaati;

	p1->bilgiGir();
	p2->bilgiGir();

	p1->bilgiGoster();
	cout<<"\n-------------------------\n";
	p2->bilgiGoster();

	//new ile oluşturulan nesneler Yok edilmeli    delete(p1);
	delete(p1);
	delete(p2);
	system("pause");

	return 0;
}
```

delete komutuyla nesne yok edilirken yıkıcısı çalışır…

PersonelIsaretci.cpp

Void İşaretçi

- ✓ **Herhangi bir tipi gösterebilir**
- ✓ *** İşareti doğrudan kullanılamaz**
- ✓ **tip dönüşümü gerektirebilir.**

```
#include <iostream>
using namespace std;

void increase (void* data, int psize)
{
  if ( psize == sizeof(char) )
  { char* pchar; pchar=(char*)data; ++(*pchar); }
  else if (psize == sizeof(int) )
  { int* pint; pint=(int*)data; ++(*pint); }
}

int main ()
{
  char a = 'x';
  int b = 1602;
  increase (&a,sizeof(a));
  increase (&b,sizeof(b));
  cout << a << ", " << b << endl;

  system("pause");
  return 0;
```

ptrint = reinterpret_cast<int*>(flovar);
ptrflo = reinterpret_cast<float*>(intvar);

ptrint = (int*)flovar;
ptrflo = (float*)intvar;

```
int main()
 {
        int intvar;          //integer variable
        float flovar;        //float variable

        int* ptrint;         //define pointer to int
        float* ptrflo;       //define pointer to float
        void* ptrvoid;       //define pointer to void

        ptrint = &intvar;    //ok, int* to int*
      // ptrint = &flovar;    //error, float* to int*
      // ptrflo = &intvar;    //error, int* to float*
        ptrflo = &flovar;    //ok, float* to float*

        ptrvoid = &intvar;   //ok, int* to void*
        ptrvoid = &flovar;   //ok, float* to void*
        return 0;
}
```

voidPointer.cpp

Örnek

Department.cpp

Kaynaklar

- ✓ Robert Lafore, Object Oriented Programming in C++, Macmillan Computer Publishing
- ✓ Deitel, C++ How To Program, Prentice Hall
- ✓ Prof. Dr. Celal ÇEKEN, Programlamaya Giriş Ders Notları
- ✓ Prof. Dr. Cemil ÖZ, Programlamaya Giriş Ders Notları