



BM102

# Algoritma ve Programlama II

**Operatör Yükleme**

# İçerik



1. Operatör Yükleme Fonksiyon Tanımı
  2. Üye Olmayan Fonksiyonlar
  3. Operatör Yüklemede Arkadaş Fonksiyon Kullanımı
  4. Girdi/Çıktı Operatörlerine Anlam Yükleme
  5. Operatör Yükleme Yapılan Operatörlere Örnekler
- Çözümlü Sorular***

# Hedefler



- Operatör yükleme kavramını anlatma
- Operatör yükleme fonksiyon tanımlama
- Üye ve üye olmayan fonksiyonlar arasındaki farkı anlatma
- Üye olmayan fonksiyon tanımlama
- Operator yükleme fonksiyonlarını arkadaş fonksiyonları olarak tanımlama
- Bir nesne için girdi ve çıktı operatörlerine yükleme yapma

# Operatör Yükleme

- *İng. Operator Overloading*
- Programlama dillerinin temel yapılarında bir çok operatör yer almaktır.
  - Matematiksel operatörler (+,--,/,\*vb.),
  - Mantıksal operatörler (!, ||, &&, vb.),
  - İlişkisel operatörler(<,>, == vb.), vb.
- Operatörlerin farklı kullanım amaçları ile yeniden tanımlanması işlemidir

# 1. Operatör Yükleme Fonksiyon Tanımı

- Operatör yükleme işlemini gerçekleştirmek için söz konusu operatörler sınıf içerisinde yeniden tanımlanır
- Operatörler tanımı operatör fonksiyonları ile yapılır

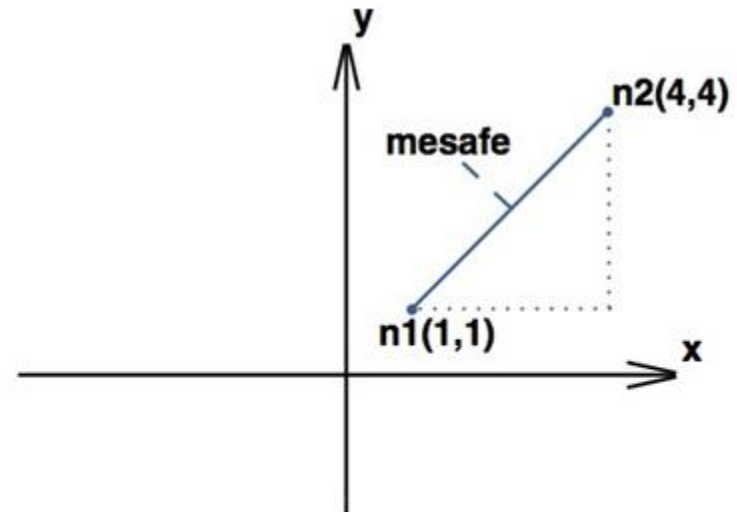
# 1. Operatör Yükleme Fonksiyon Tanımı...

## Örnek

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

+ Nokta
-x : int -y : int
+mesafe(n2 : Nokta) : double +noktaAta(x : int,y : int) : void

1 Nokta Sınıf Diyagramı



2 Koordinat Düzleminde İki Nokta Nesnesi, n1 ve n2

# 1. Operatör Yükleme Fonksiyon Tanımı...

```
//----- Nokta.h -----  
class Nokta  
{  
    public:  
        Nokta(int x=0, int y=0){noktaAta(x,y);}   
        void noktaAta(int x, int y);  
        double mesafe(const Nokta &);  
    private:  
        int x, y;  
};  
//----- Nokta.cpp -----  
#include <iostream>  
#include "Nokta.h"  
#include "math.h"  
using namespace std;  
void Nokta::noktaAta(int i, int j)           // Noktanın koordinatlarını atar  
{  
    x = i;  
    y = j;  
}  
double Nokta::mesafe(const Nokta & n)       // Mesafeyi hesaplar  
{  
    return sqrt((n.x-x)*(n.x-x)-  
                (n.y-y)*(n.y-y));  
}
```

```
//-----Uygulama.cpp -----  
#include <iostream>  
#include "Nokta.h"  
int main()  
{  
    Nokta n1(1,1), n2(4,4);  
    cout<<"Uzaklik = "<<n1.mesafe(n2)<<endl;  
    return 0;  
}
```

**Çıktı**

Uzaklik = 4.24264

# 1. Operatör Yükleme Fonksiyon Tanımı...

```
//----- Nokta.h -----
class Nokta
{
public:
    Nokta(int x=0, int y=0){noktaAta(x,y);}
    void noktaAta(int x, int y);
    double operator-(const Nokta &);    // Operatör fonksiyon prototipi
private:
    int x, y;
};

//----- Nokta.cpp -----
#include <iostream>
#include "Nokta.h"
#include "math.h"
void Nokta::noktaAta(int i, int j)    // Noktanın koordinatlarını atar
{
    x = i;
    y = j;
}
double Nokta::operator-(const Nokta & n)    // Operatör fonksiyon mesafeyi hesaplar
{
    return sqrt((n.x-x)*(n.x-x)-
                (n.y-y)*(n.y-y));
}

//-----Uygulama.cpp -----
#include <iostream>
#include "Nokta.h"
int main()
{
    Nokta n1(1,1), n2(4,4);
    cout<<"Uzaklik = "<<n1 - n2<<endl;    // - operatör fonksiyon çağırımı
    return 0;
}
```

**Çıktı**

Uzaklik = 4.24264



# 1. Operatör Yükleme Fonksiyon Tanımı...

```
//----- Nokta.h -----  
class Nokta  
{  
    public:  
        Nokta(int x=0, int y=0){noktaAta(x,y);}   
        void noktaAta(int x, int y);  
        Nokta operator+(Nokta &);  
        Nokta operator+(int);  
        void goruntule();  
    private:  
        int x, y;  
};  
//----- Nokta.cpp -----  
#include "Nokta.h"  
#include <iostream>  
using namespace std;  
void Nokta::noktaAta(int i, int j)    // Noktanın koordinatlarını atar  
{  
    x = i;  
    y = j;  
}  
Nokta Nokta::operator+(Nokta &n)// Noktaya n noktasını ekleyen operatör fonksiyon  
{  
    Nokta yeniNokta;  
    yeniNokta.x = x + n.x;  
    yeniNokta.y = y + n.y;  
    return yeniNokta;  
}
```



# 1. Operatör Yükleme Fonksiyon Tanımı...

```
Nokta Nokta::operator+(int i)    // Noktaya i değerini ekleyen operatör fonksiyon
{
    x = x + i;
    y = y + i;
    return Nokta(x,y);
}
void Nokta::goruntule()           // Noktayı yazdıran fonksiyon
{
    cout<<"("<<x<<"", "<<y<<"")"<<endl;
}
```



# 1. Operatör Yükleme Fonksiyon Tanımı...

```
//-----Uygulama.cpp -----  
#include <iostream>  
#include "Nokta.h"  
int main()  
{  
    Nokta n1(1,1), n2(4,4), n3;  
    n1.goruntule();  
    n2.goruntule();  
    n3 = n1 + n2;           // + operatör fonksiyon çağırımı  
    n3.goruntule();  
    n1 = n1 + 8;           // + operatör fonksiyon çağırımı  
    n1.goruntule();  
    n3 = n1 + n2 + 5;      // + operatör fonksiyon çağırımı  
    n3.goruntule();  
    return 0;  
}
```

## Çıktı

```
(1,1)  
(4,4)  
(5,5)  
(9,9)  
(18,18)
```

# 1. Operatör Yükleme Fonksiyon Tanımı...

- Operatör yükleme işleminde bazı kısıtlamalara dikkat etmemiz gerekir.
  - Operatörlerin standart veri tipleri (int, float, char, vb.) ile kullanımlarındaki anlamları yeniden tanımlanamaz
  - Yeni anlam kazandırma işlemi sadece C++ programlama dilinde hali hazırda tanımlı olan operatörler için yapılabilir
  - Yeni operatör tanımı yapılamaz.
  - Operatörlerin türleri değiştirilemez
  - # ve ## gibi ön işlemci (preprocessor) sembollerine de yükleme yapılamaz

# 1. Operatör Yükleme Fonksiyon Tanımı...

- Operatör yükleme işleminde bazı kısıtlamalara dikkat etmemiz gerekir...
  - Operatör yükleme işlemi aşağıdaki operatörlere uygulanamaz

**Tablo 1** Yeni Anlam Yüklenemeyen Operatörler

Operatör	İsmi
.	Üye Erişim
::	Kapsam Çözümleme
.*	Göstergeler ile Üye Erişim
?:	Üçlü Koşul
sizeof	Büyüklik

## 2. Üye Olmayan Fonksiyonlar

- Operatör fonksiyonlar, üye fonksiyon ve üye olmayan fonksiyonlar olarak tanımlanabilir. Bazı durumlarda operatör fonksiyonun üye fonksiyon olarak tanımlanması mümkün olmayabilir.
- Derleyici açısından üye ve üye olmayan fonksiyonları, operatör yükleme işleminde farklı yorumlar

Nokta  $n1(1,1)$ ,  $n2(4,4)$ ;

$n2 = n2 + 10$ ;

$n2 = 10 + n2$ ;

## 2. Üye Olmayan Fonksiyonlar...

- Eğer `operator+` üye fonksiyon olarak tanımlanmışsa, derleyici aşağıdaki şekilde yorumlar;
  - `n2.operator+(10)`
  - `10.operator+(10)`     *//geçersiz, fonksiyon ancak nesne üzerinden çağrılır*
- Eğer `operator+` üye **olmayan** fonksiyon olarak tanımlanmışsa, derleyici aşağıdaki şekilde yorumlayacaktır;
  - `operator+(n2,10)`
  - `operator+(10,10)`

## 2.Üye Olmayan Fonksiyonlar...

Nokta **operator**+(Nokta &); *// Üye fonksiyon*

Nokta **operator**+(Nokta &, Nokta &); *// Üye olmayan fonksiyon*

- Üye olmayan fonksiyon tanımı yapıldığında dikkat edilmesi gereken bir nokta vardır. Üye olmayan fonksiyonlar, ilgili sınıfın `private` üyelerine erişemeyeceği için bu üyeler için erişimci fonksiyon tanımlarının yapılması gereklidir.



Tüm üye fonksiyonlar üye olmayan fonksiyonlar olarak da tanımlanabilir ama tersi doğru değildir.



## 2. Üye Olmayan Fonksiyonlar...

```
//----- Nokta.h -----  
class Nokta  
{  
    public:  
        Nokta(int x=0, int y=0){noktaAta(x,y);}  
        void noktaAta(int x, int y);  
        void goruntule(void);  
        Nokta operator+(const int i);           // + operatör fonksiyon tanımı  
        int xAl(void){return x;}               // Erişimci fonksiyonlar  
        int yAl(void){return y;}  
        void xAta(int _x){x = _x;}  
        void yAta(int _y){y = _y;}  
    private:  
        int x, y;  
};  
//----- Nokta.cpp -----  
#include "Nokta.h"  
#include <iostream>  
using namespace std;  
void Nokta::noktaAta(int i, int j)  
{  
    x = i;  
    y = j;  
}
```



## 2. Üye Olmayan Fonksiyonlar...

```
Nokta Nokta::operator+(const int i)           // Üye operatör fonksiyonu
{
    x = x + i;
    y = y + i;
    return Nokta(x,y);
}
Nokta operator+(const int i, Nokta& n)        // Üye olmayan operatör fonksiyonu
{
    Nokta n2;
    n2.xAta(n.xAl()+i);
    n2.yAta(n.yAl()+i);
    return n2;
}
void Nokta::goruntule(void)
{
    cout<<"("<<x<<","<<y<<")"<<endl;
}
//-----Uygulama.cpp -----
#include <iostream>
#include "Nokta.h"
int main()
{
    Nokta n(5,5);
    n = n + 13;
    n.goruntule();
    n = 13 + n;
    n.goruntule();
    return 0;
}
```

**Çıktı**

(18,18)  
(31,31)

### 3. Operatör Yüklemede Arkadaş Fonksiyon Kullanımı

- Üye olmayan fonksiyonların bir sınıfın private üyelerine doğrudan erişmesini friend fonksiyonlarla olur
- Erişimci fonksiyonların eklenmesine gerek kalmaz

### 3. Operatör Yüklemede Arkadaş Fonksiyon Kullanımı...

```
//----- Nokta.h -----  
class Nokta  
{  
    public:  
        Nokta(int x=0, int y=0){noktaAta(x,y);}   
        void noktaAta(int x, int y);  
        friend Nokta operator+(const Nokta &, const Nokta &);  
        friend Nokta operator+(const int i, const Nokta&);  
        void goruntule();  
    private:  
        int x, y;  
};  
//----- Nokta.cpp -----  
#include "Nokta.h"  
#include <iostream>  
using namespace std;  
void Nokta:: noktaAta(int i, int j)  
{  
    x = i;  
    y = j;  
}
```



### 3. Operatör Yüklemede Arkadaş Fonksiyon Kullanımı...

```
Nokta operator+(const Nokta & n1, const Nokta & n2)
// n1 noktasına n2'yi ekleyen arkadaş operatör fonksiyon
{
    Nokta yeniNokta;
    yeniNokta.x = n1.x + n2.x;
    yeniNokta.y = n1.y + n2.y;
    return yeniNokta;
}
Nokta operator+(const int i, const Nokta & n)
// n noktasına i'yi ekleyen arkadaş operatör fonksiyon
{
    int x = n.x + i;
    int y = n.y + i;
    return Nokta(x,y);
}
void Nokta::goruntule()
{
    cout<<"("<<x<<"", "<<y<<"")"<<endl;
}
```



### 3. Operatör Yüklemede Arkadaş Fonksiyon Kullanımı...

```
//-----Uygulama.cpp -----  
#include <iostream>  
#include "Nokta.h"  
int main()  
{  
    Nokta n1(1,1), n2(4,4), n3;  
    n1.goruntule();  
    n2.goruntule();  
    n3 = n1 + n2;  
    n3.goruntule();  
    n2 = 13 + n2;  
    n2.goruntule();  
    n3.goruntule();  
    return 0;  
}
```

#### Çıktı

```
(1,1)  
(4,4)  
(5,5)  
(17,17)  
(5,5)
```

# 4. Girdi/Çıktı Operatörlerine Anlam Yükleme

- Temel girdi/çıktı işlemleri `>>` ve `<<` operatörleri ile gerçekleştirilir.
- Temel veri tipindeki değişkenler `cout` ve `cin` ile kullanılabilir

```
int x = 5;
cout<<x;
cin>>x
```
- Ancak bir nesne `cout` ve `cin` ile doğrudan kullanılamaz

```
Nokta n;
cout<<n;
cin>>n;
```
- Bunu sağlamak için operatör<< ve operatör>> yükleme fonksiyonlarının tanımlanması gerekir.

## 4. Girdi/Çıktı Operatörlerine Anlam Yükleme...

```
//----- Nokta.h -----
#ifndef NOKTA_H
#define NOKTA_H
#include <iostream>
using namespace std;
class Nokta
{
    double x;
    double y;
public:
    Nokta( double = 0.0, double = 0.0);
    void noktaAta(double, double);
    friend ostream& operator<<( ostream&, Nokta&);
    friend istream& operator>>( istream&, Nokta &);
};
#endif
//----- Nokta.cpp -----
#include <iostream>
#include "Nokta.h"
ostream& operator<<( ostream &os, Nokta& n)//<< operatör yükleme fonksiyonu
{
    os << "(" << n.x << "," << n.y << ")";
    return os;
}
```





## 4. Girdi/Çıktı Operatörlerine Anlam Yükleme...

```
istream& operator>>( istream &is, Nokta& n) //>> operatör yükleme fonksiyonu
{
    is >> n.x;
    is >> n.y;
    return is;
}
Nokta::Nokta(double x, double y)
{
    noktaAta(x, y);
}
void Nokta::noktaAta(double _x, double _y)
{
    x = _x;
    y = _y;
}
//-----Uygulama.cpp -----
#include <iostream>
#include "Nokta.h"
int main()
{
    Nokta n(3,4);
    cout<<n<<endl;
    cout << "Nokta giriniz: (x, y)? "<<endl;
    cin >> n;
    cout << n << endl;
    cout<<endl;
    return 0;
}
```

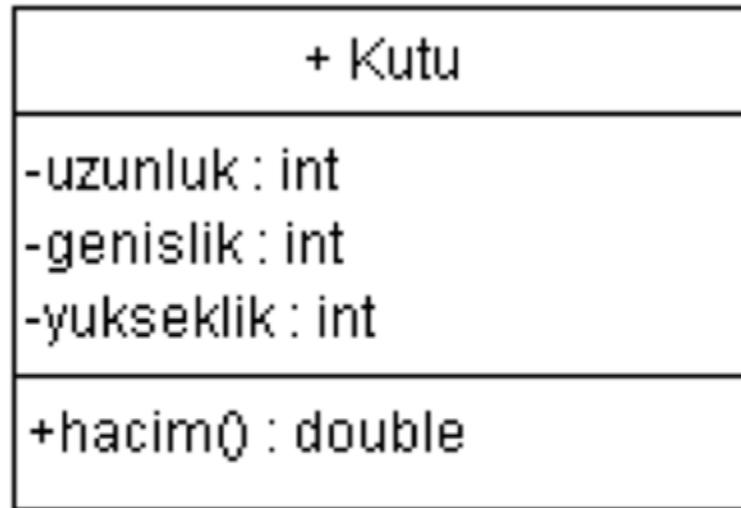
### Çıktı

```
(3,4)
Nokta giriniz: (x,y)?
5 6
(5,6)
```

## 5. Operatör Yükleme Yapılan Operatörler'e Örnekler

### > *ve* < Karşılaştırma Operatörleri

`hacim = uzunluk*yükseklik*genişlik`



**Şekil 3** Kutu Sınıf Diyagramı



## 5. Operatör Yükleme Yapılan Operatörler'e Örnekler ...

```
//----- Kutu.h -----  
class Kutu  
{  
    public:  
        Kutu(int u,int g,int y):uzunluk(u),genislik(g),yukseklk(y){}  
        friend bool operator>(const Kutu&, const Kutu&);  
        friend bool operator<(const Kutu&, const Kutu&);  
        double hacim() const;  
    private:  
        int uzunluk;  
        int genislik;  
        int yukseklik;  
};
```



## 5. Operatör Yükleme Yapılan Operatörler'e Örnekler ...

```
//----- Kutu.cpp -----
#include "Kutu.h"
#include <iostream>
using namespace std;
bool operator>(const Kutu& k1, const Kutu& k2) // >operatör yükleme fonksiyonu
{
    return k1.hacim()>k2.hacim();
}
bool operator<(const Kutu& k1, const Kutu& k2) // <operatör yükleme fonksiyonu
{
    return k1.hacim()<k2.hacim();
}
double Kutu::hacim() const
{
    return uzunluk*genislik*yukseklık;
}
//----- Uygulama.cpp -----
#include "Kutu.h"
int main()
{
    Kutu k1(15,8,4), k2(3,7,8);
    if (k1>k2)
        cout<<"Kutu-1 buyuktur"<<endl;
    else if (k1<k2)
        cout<<"Kutu-1 kucuktur"<<endl;
    else
        cout<<"Kutu buyuklukleri esittir."<<endl;
    return 0;
}
```

**Çıktı**

Kutu-1 buyuktur

## 5. Operatör Yükleme Yapılan Operatörler'e Örnekler ...

### [ ] *İndeksli Veri Erişimi Operatörü*

```
//----- UcBoyutNokta.h -----  
class UcBoyutNokta  
{  
    public:  
        UcBoyutNokta(int x, int y, int z){a[0]=x;a[1]=y;a[2]=z;}  
        int operator[](const int i);  
    private:  
        int a[3];  
};  
//----- UcBoyutNokta.cpp -----  
#include "UcBoyutNokta.h"  
int UcBoyutNokta::operator[](const int i)    // [] operatör yükleme fonksiyonu  
{  
    return a[i];  
}
```



## 5. Operatör Yükleme Yapılan Operatörler'e Örnekler ...

```
//----- Uygulama.cpp -----  
#include <iostream>  
#include "UcBoyutNokta.h"  
using namespace std;  
int main()  
{  
    UcBoyutNokta n1(13, -12, 15);  
    cout<<n1[0]<<endl;  
    cout<<n1[1]<<endl;  
    return 0;  
}
```

**Çıktı**

13  
-12

# Çözümlü Sorular

## Soru

2. Aşağıda rasyonel sayı sınıfına ait başlık dosyası verilmiştir:

```
class Rasyonel
{
    int pay;
    int payda;
    public:
        Rasyonel (int _pay,int _payda):pay(_pay),payda(_payda){}
        int payAl(){return pay;}
        int paydaAl(){return payda;}
};
```



- a. Rasyonel sınıfı içinde `>` operator yükleme üye fonksiyonu yazın. İçinde bulunan sayı parametre olarak gönderilen sayıdan büyükse `true`, değilse `false` döndürsün. ( İpucu:  $\frac{1}{4}$  sayısını 0.25'e çevirip karşılaştırabilirsiniz).



# Çözümlü Sorular...

## Cevap...

2.

```
a. //----- Rasyonel.h -----  
class Rasyonel  
{  
    int pay;  
    int payda;  
public:  
    Rasyonel (int _pay, int _payda):pay(_pay),payda(_payda){}  
    int payAl(){return pay;}  
    int paydaAl(){return payda;}  
    bool operator>(Rasyonel&);  
};  
//----- Rasyonel.cpp -----  
#include "Rasyonel.h"  
  
bool Rasyonel::operator>(Rasyonel& r)  
{  
    float a=(float)pay/(float)payda;  
    float b=(float)r.pay/(float)r.payda;  
    return a>b;  
}
```






# Çözümlü Sorular...

## Cevap

```
// ----- Uygulama.cpp -----  
#include <iostream>  
#include "Rasyonel.h"  
using namespace std;  
  
int main()  
{  
    Rasyonel r1(1,2),r2(2,9);  
    if (r1>r2) {  
        cout<<r1.payAl()<<"/"<<r1.paydaAl()<<" > ";  
        cout<<r2.payAl()<<"/"<<r2.paydaAl()<<endl;  
    }else{  
        cout<<r1.payAl()<<"/"<<r1.paydaAl()<<" <= ";  
        cout<<r2.payAl()<<"/"<<r2.paydaAl()<<endl;  
    }  
    return 0;  
}
```



# Çözümlü Sorular...

## Soru

8. Bir tamsayının basamaklarını bulmak için `TamSayi` isimli bir sınıf tanımlayınız. `TamSayi` sınıfının üyeleri aşağıda verilmiştir:

Veri Üyesi:

- `deger` : tamsayı

Üye Fonksiyonlar:

- `>>` operatör yükleme fonksiyonu : tamsayı değerini okur
- `[]` operatör yükleme fonksiyonu: parametre olarak yollanan `i` indeksini alır ve tamsayının `i`. basamak değerini döndürür (`i=0` ise en küçük basamağı döndürür). Eğer `i`. basamak yoksa `-1` döndürür. Örneğin, eğer `x` `TamSayi` tipinde bir nesne ise ve `deger` üyesinin içine `418` atanmışsa,

`x[0]` 8 döndürür,

`x[1]` 1 döndürür

`x[2]` 4 döndürür

`x[3]` -1 döndürür

`main()` fonksiyonu ve örnek çıktı aşağıda verilmiştir. Not: Çözümünüzde dizi kullanmayın.

```
int main()
{
    TamSayi nsn;
    cin>>nsn;
    cout<<"Sayinin 0. basamagi:"<< nsn[0]<<endl;
    cout<<" Sayinin 2. basamagi:"<< nsn[2]<<endl;
    return 0;
}
```

### Örnek Çıktı

Bir tamsayi giriniz: **3478**

Sayinin 0. basamagi:8

Sayinin 2. basamagi:4



# Çözümlü Sorular...

## Cevap...

8.

```
//----- Tamsayi.h -----  
#include <iostream>  
using namespace std;  
class Tamsayi  
{  
    int deger;  
public:  
    friend istream& operator>>(istream& is, Tamsayi& m);  
    int operator[](int);  
};  
//----- Tamsayi.cpp -----  
#include "math.h"  
#include "Tamsayi.h"  
using namespace std;  
int Tamsayi::operator[](int i)  
{  
    if (deger<pow(10,i))  
        return -1;  
    else  
        return deger/((int)pow(10,i))%10;  
}
```



# Çözümlü Sorular...

## Cevap

```
// ----- Uygulama.cpp -----  
#include <iostream>  
#include "Tamsayi.h"  
using namespace std;  
istream& operator>>(istream& is, Tamsayi& m)  
{  
    is>>m.deger;  
    return is;  
}  
int main()  
{  
    Tamsayi nsn;  
    cin>>nsn;  
    cout<<"Sayinin 0. basamagi:"<< nsn[0]<<endl;  
    cout<<" Sayinin 2. basamagi:"<< nsn[2]<<endl;  
    return 0;  
}
```

