BTEP243 – Ders 5
Nesne İşaretçileri

İşaretçiler, bildiğiniz gibi bir değişkenin bellekte tutuldukları yerin adresini tutarlar. Nesne işareçtileri konusundaki işaretçiler, nesnelerin bellek üzerinde bulundukları yerin adresini tutarlar.

NOT: Eğer nesnelerin üyelerine ulaşmak istiyorsak **->** işaretini kullanmalıyız.

Kullanım şekli:

```
SınıfAdı *gösterge_ismi;
```

Örnek1:

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

class Takim {
string isim; string mevki;
int no;
public:
	Takim(string,string,int); //parametreli yapıcı
	void listele();
};
Takim::Takim(string i,string m,int n)
{
	isim=i;
	mevki=m;
	no=n;
}
void Takim::listele()
{
	cout<<"Futbolcu ismi:"<<isim<<endl;
	cout<<"Oynadigi mevki:"<<mevki<<endl;
	cout<<"Numarasi:"<<no<<endl;
	cout<<"-------------------------------------"<<endl;
}

void main()
{
	Takim Turkiye[4]={Takim("Volkan Demirel","Kaleci",1),Takim("ArdaTuran","Orta Saha",10),
Takim("Burak Yilmaz","Forvet",9),Takim("Gokhan Gonul","Defans",7)};

	Takim *A=Turkiye;
	for(int i=0; i<4; i++)
	{
		A->listele();
		A++;
}  system("pause");    }
```

Örnek2:

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

```
class Takim {
string isim,mevki;
int no;
public:
	Takim();
	void listele();
};

Takim::Takim() //Parametresiz yapıcı. Veriler klavyeden alınıyor...
{
	cout<<"Futbolcu ismi:";
	cin.ignore();
	getline(cin,isim);	cout<<"Mevki ismi:";
	cin.ignore();
	getline(cin,mevki);
	cout<<"Numarasi:";
	cin>>no;
}

void Takim::listele()
{
	cout<<"Futbolcu ismi:"<<isim<<endl;
	cout<<"Mevki ismi:"<<mevki<<endl;
	cout<<"Numarasi:"<<no<<endl;
	cout<<"---------------------------------------"<<endl;
}

void main()
{
	Takim Turkiye[4];
	Takim *A=Turkiye;
	for(int i=0; i<4; i++)
	{
		A->listele();
		A++;
}
system("pause");
}
```

Dinamik Bellek Yönetimi

Şu ana kadar küçük boyutta değişkenler kullandık ancak eğer belleğin daha büyük boyuttaki değişkenler için kullanımı gerekecekse dinamik bellek yönetimini kullanmak daha pratik olacaktır.
Bu nedenle **new** ve **delete** operatörlerini kullanacağız.

NEW Operatörü

Bellekten yer ayırmak istendiği zaman **"new"** operatörü kullanılır. Bir tamsayı gösteren işaretçi için bir tamsayılık yer aşağıdaki şekilde bellekte kendisine yer edinir.

```
int *sayi;
sayi=new int;
```

Eğer değişkene başlangıç değeri de vermek istiyorsak;

Sayi=new int(10); //sayi değişkenine 10 değerini atamış oluruz.

Eğer bellekte bir dizi oluşturacak şekilde yer ayırmak istiyorsak:

int *dizi;
dizi=new int[3]; //bellekte 3 elemanlı bir dizi için yer ayırır ve bellekte bu dizinin bulunduğu yerin başlangıç adresini dizi işaretçisine atar. Ancak dizilerdeki elemanlara başlangıç adresi veremeyiz.

DELETE Operatörü
"**new**" operatörü ile bir değişken için yer ayırdığımızı varsayalım. Ayırdığımız bu yeri geri vermek istiyorsak **delete** operatörünü kullanırız.

int *sayi;
sayi=new int; ⇨ delete sayi

int *dizi;
dizi=new int[3]; ⇨ Delete [ ] dizi;

Örnek1:

```
#include <iostream>
using namespace std;
void main()
{
	int *p;
	p=new int;
	*p=1000;
	cout<<"Pdeki tamsayi:"<<*p<<endl;

	delete p;

	system("pause");
}
```

Örnek2:

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

void main()
{
int *p = new int (5) ;
long *longPtr;
longPtr=new long(100000);
cout<<longPtr<<"   "<<*longPtr;
*longPtr  *= 2;
cout<<endl<<longPtr<<"   "<<*longPtr;
cout<<endl<<p<<"   "<<*p;
*p *=2;
```

```
cout<<endl<<p<<"   "<<*p;
system("pause");
}
```
Çıktı:
```
00494468   100000
00494468   200000
00494420   5
00494420  10
```

Örnek3:
```
#include <iostream>

using namespace std;
class dinamik{
  int x,y;
public:
  void ata(int sayi1,int sayi2)
  {
        x=sayi1;
        y=sayi2;
  }
  int carp()
  {
        return x*y;
  }
};
void main()
{
  dinamik *p;
  p=new dinamik;
  p->ata(5,6);
  cout<<"Carpimlari:"<<p->carp()<<endl;

  delete p;
  system("pause");
}
```

Örnek4:

```
//1 float ,long va char için yer ayır. Bunlara sırayla deger ver ve delete ile ekranda //gosterdikten sonra alanı bosalt
#include<iostream>
using namespace std;
void main()
{
        float *f;
        long *l;
        char *c;

        f=new float;
        l=new long;
```

```
	c=new char;

	*f=10.2;
	*l=10000;
	*c='A';


	cout<<*f<<endl<<*l<<endl<<*c<<endl;

	delete f;
	delete l;
	delete c;

	system("pause");
}
```

Örnek5:

```
//kisi ismi ve telefon numarasını sınıf yaratarak parametre olarak al.
//bellekten nesne için yer ayır sonra isim ve noyu goster.
#include<iostream>
using namespace std;

class telefon
{
	char isim[20];
	char numara[20];
public:
	void ata(char *is, char *no)
	{
		strcpy(isim,is);
		strcpy(numara,no);
	}
	void goster()
	{
		cout<<isim<<":"<<numara<<endl;
	}
};
void main()
{
	telefon *p;
	p=new telefon;
	p->ata("Sebnem","03926301677");
	p->goster();

	delete p;
	system("pause");
}
```

Örnek6:

```
//kisi ismi ve telefon numarasını sınıf yaratarak parametre olarak al.
//bellekten nesne için yer ayır sonra isim ve noyu goster.
#include<iostream>
using namespace std;

class telefon
{
	char isim[20];
	char numara[20];
public:
	void ata()
	{
		cout<<"Isim gir:";
		cin>>isim;
		cout<<"Telefon no gir:";
		cin>>numara;
	}
	void goster()
	{
		cout<<isim<<":"<<numara<<endl;
	}
};
void main()
{
	telefon *p;
	p=new telefon[5];
	for(int i=0;i<5;i++)
		p[i].ata();

	for(int i=0;i<5;i++)
		p[i].goster();

	delete []p;
	system("pause");
}
```

Örnek7:

```
#include <iostream>
using namespace std;
class nokta {
double x, y;
public:
 nokta()
 { x=1;y=2;}
 void goster ()
 {  cout<<"("<<x<<","<<y<<")"<<endl; }
 void ata ( double u , double v)
 {  x = u ; y = v ;          }
  ~nokta()
 { cout<<"\nObjenin değerleri: "<<x<<" "<<y<<" silindi"; }
```

```
};
 void main()
 {
 nokta *a;
 cout<<a<<endl;
 a=new nokta[4];
 int k;
 for (k=0;k<4;k++)
 a[k].goster();
delete [] a;
 }
```