BTEP243 – Ders 7
Sınıfların Sınıfları Kullanma Biçimi

Sınıfların diğer sınıfları kullanma şekilleri:

1. **Veri elemanı olarak kullanma** (**composition**): Bir sınıfın başka bir sınıf türünden veri elemanına sahip olması durumunda eleman olan sınıf nesnesinin ömrü, elemana sahip sınıf nesnesinin ömrüyle ilgilidir. Yani, eleman olan sınıf nesnesi, elemana sahip sınıf nesnesi yaratıldığında yaratılır ve o nesne yok edildiğinde yok edilir.

Örnek1: (Sınıf Birleştirme (Composition))

//dgunu.h

```
class dogumgunu{
private:
        int gun;
        string ay;
        int yil;
public:
        dogumgunu(int g,string a,int y)
        {
                gun=g,
                ay=a;
                yil=y;
        }
        void dgunuGoster()
        {
                cout<<gun<<"/"<<ay<<"/"<<yil<<endl;
        }
};
```

//ogrenci.h

```
class ogrenci{
private:
        string isim;
        dogumgunu dgunu;
public:
        ogrenci(string isim,dogumgunu dg):dgunu(dg)
        {
                this->isim=isim;
        }
        void ogrenciGoster()
        {
                cout<<isim<<" ";
                dgunu. dgunuGoster ();
                cout<<" tarihinde doğdu."<<endl;
        }
};
```

//ogrenciDgunu.cpp

```
#include<iostream>
#include<string>
using namespace std;
#include"dgunu.h"
#include"ogrenci.h"
```

```
void main()
{
	dogumgunu dgunu(23,"Ocak",1900);
	ogrenci ben("Sebnem",dgunu);
	ben. ogrenciGoster ();
	system("pause");
}
```

Ekran Çıktısı:

Sebnem 23/Ocak/1900 tarihinde doğdu.

<u>Örnek 2:</u> (Sınıf Birleştirme (Composition) Örneği-- Varsayılan Yapıcı)

//bolum.h

```
class bolum{
private:
	int bolno;
	string boladi;
public:
	bolum()
	{
		bolno = 10;
		boladi = "BTEP";
	}
	void bolumGoster()
	{
		cout << "Bolum no:" << bolno << endl;
		cout << "Bolum Adi:" << boladi << endl;
	}
};
```

//ogrenci.h

```
class ogrenci{
private:
	int ogrno;
	string ogradi;

	bolum ogrbol; //bolum sınıfının objesini yaratarak, ogrenci ve bolum sınıflarını
			//ilişkilendirmiş olduk.
public:
	ogrenci()
	{
		ogrno = 101;
		ogradi = "ibrahim";
	}
	void ogrencigoster()
	{
		cout << "Ogrenci No:" << ogrno << "\t"
		     << "Ogrenci Adi:" << ogradi << endl;
		ogrbol.bolumGoster();
	}
};
```

//ogrencibolum.cpp

```
#include<iostream>
#include<string>
using namespace std;
#include"bolum.h"
#include"ogrenci.h"
void main()
{
	ogrenci ogr1;
	ogr1.ogrencigoster();
	system("pause");
}
```

<u>Örnek 3:</u> (Sınıf Birleştirme (Composition) Örneği – Parametreli Yapıcı)

//bolum.h

```
class bolum{
private:
	int bolno;
	string boladi;
public:
	bolum(int bno,string badi)
	{
		bolno = bno;
		boladi = badi;
	}
	void bolumGoster()
	{
		cout << "Bolum no:" << bolno << endl;
		cout << "Bolum Adi:" << boladi << endl;
	}
};
```

//ogrenci.h

```
class ogrenci{
private:
	int ogrno;
	string ogradi;
	bolum ogrbol; //bolum sınıfının objesini yaratarak, ogrenci ve bolum sınıflarını
	//ilişkilendirmiş olduk.
public:
	//ogrenci(int ono, string adi, int bno, string boladi) :ogrbol(bno,boladi)
	ogrenci(int ono, string adi, bolum bol) :ogrbol(bol)
	{
		ogrno = 101;
		ogradi = "ibrahim";
	}
	void ogrencigoster()
	{
		cout << "Ogrenci No:" << ogrno << "\t"
			<< "Ogrenci Adi:" << ogradi << endl;
		ogrbol.bolumGoster();
```

```
        }
};
```

//ogrencibolum.cpp

```
#include<iostream>
#include<string>
using namespace std;
#include"bolum1.h"
#include"ogrenci1.h"
void main()
{
        //ogrenci ogr1(101,"Ibrahim",10,"BTEP");
        bolum bol(10, "BTEP");
        ogrenci ogr(101, "Ibrahim", bol);
        ogr.ogrencigoster();
        system("pause");
}
```

Örnek 3- Çözüm 1

//uye.h

```
class uye{
  private:
        string isim;
  public:
        uye(string isim)
    {
        this->isim=isim;
    }
   void UyeGoster()
    {
        cout<<"Uye ismi:"<<this->isim;
    }
};
```

//pgrup.h

```
#include "member.h"
class pgrup{
  private:
        int grupno;
        uye uye1, uye2;
         static int sayac;
  public:
  team(uye u1, uye u2):uye1(u1),uye2(u2)
    {
        grupno=sayac;
        sayac++;
    }
   void pgrupGoster()
    {
        cout<<"Grup No:"<<grupno<<endl;
        uye1. UyeGoster (); cout<<endl;
        uye2. UyeGoster (); cout<<endl;
    }   }; int pgrup::sayac=18;
```

//proje.h

```
class proje{
  private:
        int pno;
         string padi;
          pgrup pgrup1;
  public:
        proje(int pn,string pa, pgrup pg): pgrup (pg)
        {
              pno=pn;
              padi=pa;
         }
         void projeGoster()
         {
              cout<<"Projet Numarasi ve Adi:"<<pno<<" "<<padi<<endl;
              pgrup1. pgrupGoster ();
          }
};
#include<iostream>
#include<string>
using namespace std;
#include "uye.h"
#include "pgrup.h"
#include "proje.h"
void main()
{
  uye uye1("Ayse"), uye2("Ali"), uye3("Sebnem"), uye4("Sinan");
  pgrup grup1(uye1,uye2), grup2(uye3,uye4);
  projet proje1(1,"Yurtlar",grup1), proje2(2,"Stok Kontrol",grup2);
  proje1. projeGoster ();
  proje2. projeGoster ();
  system("pause");
}
```

<u>Örnek 3- Çözüm 2</u>

//uye.h

```
class uye{
  private:
        string isim;
  public:
        uye(string isim)
    {
        this->isim=isim;
    }
   void UyeGoster()
    {
        cout<<"Uye ismi:"<<this->isim;
    }
};
```

//pgrup.h

```
#include<string>
#include "mem1.h"
class pgrup{
  private:
        int grupno;
        uye uye1, uye2;
        static int sayac;
  public:
        pgrup(string uyeadi1, string uyeadi2):uye1(uyeadi1),uye2(uyeadi2)
        {
                grupno=sayac;
                sayac++;
        }
        void pgrupGoster()
        {
                cout<<"Grup No:"<<grupno<<endl;
                uye1. UyeGoster (); cout<<endl;
                uye2. UyeGoster (); cout<<endl;
         }
  }; int pgrup::sayac=18;
```

//proje.h

```
class proje{
  private:
        int pno;
         string padi;
          pgrup pgrup1;
  public:
        project(int pn, string pa,string uye1,string uye2): pgrup1 (uye1,uye2)
        {
                pno=pn;
                padi=pa;
          }
          void projeGoster()
          {
                cout<<"Projet Numarasi ve Adi:"<<pno<<" "<<padi<<endl;
                pgrup1. pgrupGoster ();
           }
};
```

//main

```
#include<iostream>
#include<string>
using namespace std;
#include "uye.h"
#include "pgrup.h"
#include"proje.h"
void main()
{
  project proje(1,"Yurtlar","Ahmet","Ayse"), proje2(2,"Stok Kontrol","Sebnem","Sinan");
  proje1. projeGoster ();
  proje2. projeGoster ();
  system("pause"); }
```

2. **Türetme (Kalıtım) ilişkisi ile kullanma** (**inheritance**): Kalıtım, var olan bir ana sınıfın özelliklerinin yeri geldiğinde değiştirilerek yeri geldiğinde ise yeni özellikler eklenilerek bir alt sınıfa aktarılmasıdır. Yukarıdan aşağıya doğru hiyerarşik bir yapısı bulunmaktadır. Bu nedenle en üst kademedeki ana sınıfa "**temel (ana) sınıf**", temel sınıftan alt sınıftaki sınıflara ise "**türetilmiş sınıf**" denir.

Ana sınıftan bir türetilmiş sınıf tanımlarken öncelikle türetilmiş sınıfın adı yazılır ve devamında ": " ve erişim türü ile ana sınıfın adı yazılır. Erişim türü **public, private** veya **protected** olabilir.

class *türetilen_sınıf* : erişim_türü ana_sınıf { }

Erişim türü her zaman yazılmak zorunda değildir. Fakat biz yazılımın anlaşılırlığı açısından erişim türünü yazmaya özen göstermeliyiz. Sınıf türetme işleminde varsayılan erişim türü **private**'dir.

```
class point { // temel sınıf
int x,y;
public:
void setPoint(int X,int Y)
{
        x=X;
        y=Y;
}
};
class dikdortgen : public point { // turetilmis sinif
        int genislik, uzunluk ;
public:
        void setBoyut(int g,int u)
        {
                genislik=g;
                uzunluk=u;
        }
};
class daire : public rectangle { // turetilmis sinif
public:
        void setycap(int r)
        {
                uzunluk=genislik=r;
        }
};
```

C++ dilinde yazılan kodun yeniden kullanılabilir olmasını sağlayan mekanizma kalıtımdır. Yeniden kullanılabilirlikten, bir sınıfın alınıp bir başka yazılım uygulamasında da (aynen yada değişikliklerle birlikte) kullanılabilmektedir.

Hatırlatma: Bir sınıf üyesi (sınıf içerisindeki) diğer tüm üyelere erişebilir. O sınıftan bir nesne ise sadece public ile tanımlı üyelere erişebilir. Kalıtım mekanizmasında, türetilmiş sınıf üyelerinin, temel sınıf üyelerine erişimi nasıl denetlenebilir?

Kural: türetilmiş sınıf üyeleri temel sınıfın **public** ve **protected** ile tanımlanmış üyelerine erişebilir.

Erişim	Kendi sınıfından erişim	Türetilmiş sınıftan erişim	Dışarıdan erişim
private	Evet	Hayır	Hayır
protected	Evet	Evet	Hayır
public	Evet	Evet	Evet

Private Sınıf Türetme

Protected Sınıf Türetme

Public Sınıf Türetme

Genel olarak, üyeleri private tanımlamak uygun olacaktır. Böylelikle dışarıdan bir fonksiyonun yanlışlıkla üyenin değerini değiştirmesi olasılığı ortadan kaldırılmış olur. Temel sınıf tasarlanırken olabildiğince protected kullanılmasından kaçınılmalıdır. Yeni sınıflar kalıtım yoluyla türetilerek genişletildikçe, üst sınıfların temel sınıf üyelerine erişimi (karmaşıklığı) önlenmiş olur. Böylelikle daha kararlı ve güvenilir sınıflar gerçekleştirilebilir.

Kalıtım ile Aktarılamayan Fonksiyonlar

Temel sınıfta tanımlı bir fonksiyon, eğer işlev yükleme yapılmamış ise, otomatik olarak türetilen sınıf üyelerinin kullanımına aktarılır. Ancak bazı özel fonksiyonlar, kalıtım ile türetilmiş sınıfa aktarılmazlar:

- **Yapıcı Fonksiyonlar:**Temel sınıfın yapıcı fonksiyonu türetilmiş sınıfın yapıcı fonksiyonu değildir.
- **Yıkıcı Fonksiyonlar:**Temel sınıfın yıkıcı fonksiyonu türetilmiş sınıfın yıkıcı fonksiyonu değildir.

Yapıcı Fonksiyonlar ve Türetim

Türetilmiş sınıftan bir nesne yaratıldığında, temel sınıfa ait yapıcı fonksiyon türetilmiş sınıfa ait yapıcı fonksiyondan önce çağrılır. Temel sınıf üyeleri türetilmiş sınıfın bir alt parçası olduğundan üst parça oluşturulmadan önce alt parçalara ait üyelerin yapılandırılması zorunluluğu vardır.

Eğer temel sınıf, parametre alan bir yapıcı fonksiyona sahip ise türetilmiş sınıfa ait yapıcı fonksiyon, temel sınıf yapıcı fonksiyonunu, uygun parametreler ile çağıracak bir yapıcıya sahip olmalıdır.

Yıkıcı Fonksiyonlar ve Kalıtım

Yıkıcı fonksiyonlar nesnenin erişimi dışına çıkıldığında otomatik olarak çağırılırlar. Kalıtım ile türetilmiş sınıflarda yıkıcı fonksiyonların çağırılış sırası yapıcı fonksiyonların çağırılış sırasının tersi şeklindedir. Bu durumda ilk olarak türetilmiş sınıfın yıkıcı fonksiyonu çağırılacaktır.

PUBLIC ile sınıf türetme

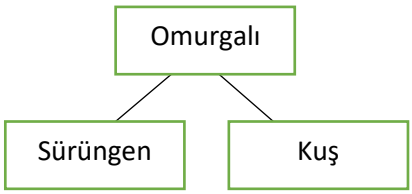
public olarak sınıf türetme işleminde, erişim türü olarak **public** anahtar sözcüğünü kullanırız.

class *türetilen_sınıf* : **public** ana_sınıf { }

Bir sınıf bir başka sınıftan ***public*** olarak türetilmişse, türetildiği ana sınıfın, public bütün elemanları türetilmiş sınıfta ***public*** elemanlar olur. Türetilmiş sınıfın bütün nesneleri ana sınıfın public elemanlarına erişebilir.

Burada en önemli nokta; türetilmiş sınıf nesnelerinin ana sınıfın ***private*** elemanlarına erişimlerinin mümkün olmamasıdır.

Örnek1:




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

class omurgali
{
public:
	int yas;
	string ad;

	void goruntule()
	{
		cout<<"Omurgalinin yasi:"<<yas<<endl;
		cout<<"Omurgalinin adi:"<<ad<<endl;
	}
};
class surungen:public omurgali
{
public:
	surungen(int y,string a)
	{
		yas=y;
		ad=a;
	}
};
class kus:public omurgali
{
public:
	kus(int y, string a)
	{
		yas=y;
		ad=a;
	}
```

```
};

void main()
{
	kus k(3,"tweety");
	surungen s(10,"kertenkele");
	k.goruntule();
	s.goruntule();

	system("pause");
}
```

PRIVATE ile sınıf türetme

Ana sınıftan public olarak sınıf türetebildiğimiz gibi private bir sınıf da türetebiliriz. public türetmede ana sınıfın public üyeleri türetilmiş sınıfının da public üyeleri oluyordu.
Private sınıf türetmede ise, ana sınıfın public üyeleri türetilmiş sınıfın private üyeleri haline gelir.

Burada dikkat edilmesi gereken nokta; sınıf türetme işlemi public de olsa private de olsa, ana sınıfın private üyelerine türetilmiş sınıf nesnelerinin erişemeyeceğidir.

Örnek 2:

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

class x
{
public:
string a,b;
x()
{ }

};
class y:private x{
public:
	y(string x,string y)
	{
		a=x; b=y;
	}
	void goruntule()
	{
		cout<<a<<"\t"<<b<<endl;
	}
};
void main()
{
	y kisi("Sebnem","Coban");
	kisi.goruntule();

	system("pause");
}
```

Yukarıdaki örnekte ana sınıfımız olan “x” sınıfında private olarak bir “y” sınıfı türettik. “x” sınıfının ***protected*** elemanları olan string türünden a ve b değişkenleri “y” sınıfının private elemanları haline geldi.
main içerisinde kisi adında bir nesne oluşturduk ve yapıcı fonksiyona parametre olarak iki string verdik. Yapıcı fonksiyon, bu stringleri “x” sınıfından private olarak aldığı a ve b elemanlarına atadı.
kisi.goruntule() fonksiyonunu çağırarak a ve b elemanlarını ekrana yazdırdık. Bu kullanımda bir hata vermedi. Çünkü kişi nesnesi, dahil olduğu sınıfın eleman fonksiyonları yardımıyla private öğelere erişebilir.

Örnek3:

```
#include <iostream>
#include <string>
using namespace std;
class dortgen{
public:
	int kenar1,kenar2;
	dortgen()
	{};
	void alan()
	{
		cout<<"Dortgenin Alani:"<<kenar1*kenar2<<endl;
	}
};

class kare:private dortgen
{
public:
	kare(int x)
	{
		kenar1=x;
		kenar2=x;
	}
	void karealan()
	{
		alan();
	}
};
void main()
{
	kare nesne(4);
	nesne.karealan();
	system("pause");
}
```

PROTECTED Elemanlar

Türetilmiş sınıfın, ana sınıfın private öğrelerine erişmesini istediğimizde **protected** elemaları kullanacağız. Protected elemanlar bulunduğu sınıf için private olmasına rağmen diğer sınıflar için erişilebilirler.

Ana sınıfta protected olarak tanımlanmış bir eleman, public bir türetmede türetilmiş sınıfın da protected elemanıdır. private türetmede ise ana sınıfın protected üyeleri türetilmiş sınıfın private üyeleridir. Her iki durumda da ana sınıfın protected üyesine türetilmiş sınıfın eleman fonksiyonları ile erişilebilir.

Örnek4:

```
#include <iostream>
#include <string>
using namespace std;
class okul{
protected:
	string ad;
public:
	okul() {};
	okul(string x)
	{
		ad=x;
	}
	string ad_dondur()
	{
		return ad;
	}
};
class ogrenci:public okul{
	string soyad;
	int numara;
public:
	ogrenci() {};
	ogrenci(string a, string s,int no)
	{
		ad=a;
		soyad=s;
		numara=no;
	}
	void goster()
	{
		cout<<"Ad:"<<ad<<endl;
		cout<<"Soyad:"<<soyad<<endl;
		cout<<"Numara:"<<numara<<endl;
		cout<<"----------------------"<<endl;
	}
};
void main()
{
	okul uni("Dogu Akdeniz Universitesi");
	ogrenci ogr1("Sebnem","Coban",99), ogr2("Ahmet","Mehmet",100);
	ogr1.goster();
	ogr2.goster();
	cout<<uni.ad_dondur()<<endl;
	system("pause");
}
```

Örnek5:

```
#include <iostream>

using namespace std;
class A
{
```

```
protected:
	int x,y;
public:
	A() {};
};
class B:public A{
public:
	B(int sayi1,int sayi2)
	{
		x=sayi1;
		y=sayi2;
	}
	void goruntule()
	{
		cout<<x<<endl;
		cout<<y<<endl;
		cout<<"--"<<endl;
	}
	void degistir(int a,int b)
	{
		x=a; y=b;
	}
};

void main()
{
	B nesne1(2,3),nesne2(10,20);
	nesne1.goruntule();
	nesne1.degistir(3,2);
	nesne2.goruntule();
	nesne1.goruntule();
	system("pause");
}
```

Yukarıdaki örnekte A ve B sınıfları arasında public erişim sağlandığı için main ( ) fonksiyondan metotlara erişmede herhangi bir hata oluşmadı.
Çünkü B sınıfı A sınıfının tüm elemanlarını public özellik olarak alır.

Örnek6:

```
#include <iostream>

using namespace std;
class A
{
protected:
	int x,y;
public:
	A() {};
	void goruntule()
	{
		cout<<x<<endl;
```

```
		cout<<y<<endl;
		cout<<"--"<<endl;
	}
};
class B:private A{
public:
	B(int sayi1,int sayi2)
	{
		x=sayi1;
		y=sayi2;
	}

	void degistir(int a,int b)
	{
		x=a; y=b;
	}
};

void main()
{
	B nesne1(2,3),nesne2(10,20);
	nesne1.goruntule();
	nesne1.degistir(3,2);
	nesne2.goruntule();
	nesne1.goruntule();
	system("pause");
}
```

Yukarıdaki örnekte A ve B sınıfları arasında private erişim sağlandığı için main() fonksiyondan metotlara erişmede hata yaşandı.
Çünkü B sınıfı A sınıfının tüm elemanlarını private özellik olarak alır. Böylece main() fonksiyondan herhangi bir sınıfın private özelliklerine erişim mümkün değildir.

PROTECTED ile Sınıf Türetme

public ve private erişim türlerinde olduğu gibi protected olarak da sınıf türetilebilir. ***Public*** ile sınıf türetme işleminde ana sınıfın ***public*** elemanlarının türetilmiş sınıfın ***public*** elemanları haline gelir.
Ayni şekilde ***private*** sınıf türetmede ana sınıfın ***public*** veya ***protected*** elemanlarının türetilmiş sınıfın ***private*** elemanları haline gelir.
protected ile sınıf türetmede ise ana sınıfın ***public*** veya ***protected*** elemanları türetilmiş sınıfın ***protected*** elemanları haline gelir.

Örnek 7:

```
#include <iostream>
#include <string>
using namespace std;
class calisan{
protected:
	string ad,soyad;
public:
	void isim(string a, string s)
```

```
	{
		ad=a;
		soyad=s;
	}
};
class muhendis:protected calisan
{
	int maas;
public:
	muhendis(){};
	void muh_maas(int m)
	{
		maas=m;
	}
	void erisim(string a,string b)
	{
		isim(a,b);
	}
	friend void listele(muhendis &);
};
void listele(muhendis &x)
{
	cout<<"Ad:"<<x.ad<<endl;
	cout<<"Soyad:"<<x.soyad<<endl;
	cout<<"Maas:"<<x.maas<<endl;
	cout<<"--------------------"<<endl;
}

void main()
{
	muhendis kisi[3];
	kisi[0].erisim("Engin","Bilen");
	kisi[1].erisim("Dicle","Bilen");
	kisi[2].erisim("Firat","Bilen");
	kisi[0].muh_maas(2000);
	kisi[1].muh_maas(1500);
	kisi[2].muh_maas(3000);
	for(int i=0;i<3;i++)
		listele(kisi[i]);
	system("pause");
}
```

Örnek8: (Lab Çalışması)

```
#include<string>
using namespace std;
class calisan{
protected:
	int calisanNo;
	char ad[15];
public:
	calisan()
```

```
	{
		cout<<"Calisan no:"; cin>>calisanNo;
		cout<<"Calisan Ad:"; cin>>ad;
	}
	calisan(int cno,char*a)
	{
		calisanNo=cno;
		strcpy(ad,a);
	}
	void setCalisanNo(int cno)
	{
		calisanNo=cno;
	}
	void setCalisanAd(char*a)
	{
		strcpy(ad,a);
	}
	int getCalisanNo()
	{
		return calisanNo;
	}
	char* getCalisanAd()
	{
		return ad;
	}
};
class memur:public calisan{
private:
	float maas;
public:
	memur()
	{
		cout<<"Maas gir:"; cin>>maas;
	}
	memur(int cno,char*a,float m):calisan(cno,a)
	{
		maas=m;
	}
	float getmaas()
	{
		return maas;
	}
	void setmaas(float m)
	{
		maas=m;
	}
	void MemurGoster()
	{
		cout<<"Memurun numarasi:"<<getCalisanNo()<<endl;
		cout<<"Memurun adi:"<<getCalisanAd()<<endl;
		cout<<"Memurun maasi:"<<getmaas()<<endl;
		cout<<"-------------------------------------"<<endl;
```

```
	}
};

class isci:public calisan{
private:
	float saatucreti;
	float calismaSaati;
public:
	isci()
	{
		cout<<"Iscinin calisma saati:"; cin>>calismaSaati;
		cout<<"Iscinin saat basi ucreti:"; cin>>saatucreti;
	}
	isci(int cno,char*a,float su,float cs):calisan(cno,a)
	{
		saatucreti=su;
		calismaSaati=cs;
	}
	float getisciMaas()
	{
		return (saatucreti*calismaSaati)*4;
	}
	void isciGoster()
	{
		cout<<"Iscinin numarasi:"<<getCalisanNo()<<endl;
		cout<<"Iscinin adi:"<<getCalisanAd()<<endl;
		cout<<"Iscinin maasi:"<<getisciMaas()<<endl;
		cout<<"-----------------------------------"<<endl;
	}
};

#include <iostream>
#include "calisan.h"
using namespace std;

void main()
{
	cout<<"Memur bilgilerini giriniz:"<<endl;
	memur m1, m2(12,"Ahmet",5000);

	cout<<"\nIsci bilgilerini giriniz:"<<endl;
	isci i1,i2(102,"Ali",30,35.25);

	m1.MemurGoster();
	m2.MemurGoster();

	i1.isciGoster();
	i2.isciGoster();

	system("pause");
}
```

Program Çıktısı:

```
/*
Memur bilgilerini giriniz:
Calisan no:10
Calisan Ad:Mustafa
Maas gir:4500

Isci bilgilerini giriniz:
Calisan no:101
Calisan Ad:Ayse
Iscinin calisma saati:32
Iscinin saat basi ucreti:45.25
Memurun numarasi:10
Memurun adi:Mustafa
Memurun maasi:4500
--------------------------------------
Memurun numarasi:12
Memurun adi:Ahmet
Memurun maasi:5000
--------------------------------------
Iscinin numarasi:101
Iscinin adi:Ayse
Iscinin maasi:5792
----------------------------------
Iscinin numarasi:102
Iscinin adi:Ali
Iscinin maasi:4230
----------------------------------
Press any key to continue . . .
```