C++ 2.

Nesneye Yönelik Programlama

Şevket Umut ÇAKIR

PAÜ

21 Nisan 2020

Anahat

Sınıf Oluşturma

- Sınıf oluşturmak için `class` anahtar kelimesi kullanılır.
- Elemanların erişimi `public`, `private` ve `protected` etiketleri içinde grup halinde bulunur.
- Sınıf tanımından sonra `;` sembolü bulunur.

Sınıf Oluşturma

```
class SınıfAdı {
        ErişimŞekli: //private, public veya protected
        Uye değerler; //kullanılacak değişkenler
        Uye fonksiyonlar(); //kullanılacak fonksiyonlar
}; //sınıf ismi ; ile biter
```

Sınıf Örneği

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

class Araba
{
    public:
        string marka;
        string model;
        int modelyili;
        Araba():marka("Tofaş"), model("Şahin"), modelyili(1995) {}
↪   //Varsayılan yapıcı
        Araba(string marka, string model, int modelyili) {
            this->marka = marka;
            this->model = model;
            this->modelyili = modelyili;
        }
        void yazdir() {
            cout << marka << " " << model << " " << modelyili << endl;
        }
};
```

Sınıf Kullanımı

```
int main(int argc, char* argv[])
{   //Farklı kullanımları
    Araba a1;
    a1.marka = "Hyundai";
    a1.model = "Accent";
    a1.modelyili = 2010;
    a1.yazdir();
    Araba a2("Toyota", "Corolla", 1998);
    a2.yazdir();
    Araba a3;
    a3.yazdir();
    return 0;
}
```

Fonksiyon Tanımlanması

- Üye fonksiyonlar sınıfın içinde tanımlanabileceği gibi(inline), sınıf dışında, hatta başa bir dosyada, tanımlanabilir.
- Bunun için kapsam çözümleme operatörü **::** kullanılmaktadır.
- C++ nesneleri işaretçiler yardımıyla da oluşturulabilir.

Fonksiyon Tanımlanması

```
        Araba():marka("Tofaş"), model("Şahin"), modelyili(1995) {}
↪   //Varsayılan yapıcı
        void yazdir();
};

void Araba::yazdir() {
    cout << marka << " " << model << " " << modelyili << endl;
}

int main(int argc, char* argv[])
{
    Araba *a1=new Araba();
    a1->yazdir();
    return 0;
}
```

Yıkıcılar(Destructors)

- Yıkıcılar tahsis kaldırma işemleri için kullanılır.
- C++ dilinde bellek yönetimi programcı kontrolündedir.
- Sınıfın metotları içinde **heap** alanında oluşturulan tahsis işlemleri yıkıcılar ile kaldırılabilir.

Yıkıcı Örneği

```
class Dizi {
    public:
        Dizi(int boyut) {//Constructor
            this->boyut = boyut;
            bellek = new int[boyut];//Yeni dizi oluşturma
            for (int i = 0; i < boyut; i++)
                bellek[i] = i;//İlk değerler
        }
        ~Dizi(){
            cout << "Yıkıcı çalışıyor. Boyut: " << boyut << endl;
            delete bellek;
        }
        int al(int eleman) {//İstenilen konumdaki elemanı verir
            return bellek[eleman];//Hataya açık
        }
    private:
        int * bellek;
        int boyut;
};
```

Yıkıcı Kodu Kullanım

```
int main(int argc, char const *argv[])
{
    for (int i = 5; i < 10; i++) {
        Dizi a(i);
        cout << a.al(9-i) << endl;
    }
    return 0;
}
```

Yıkıcı Kodu Ekran Çıktısı

```
4
Yıkıcı çalışıyor. Boyut: 5
3
Yıkıcı çalışıyor. Boyut: 6
2
Yıkıcı çalışıyor. Boyut: 7
1
Yıkıcı çalışıyor. Boyut: 8
0
Yıkıcı çalışıyor. Boyut: 9
```

İşaretçi Kullanımı

```
#include <iostream>
using namespace std;
class Ogrenci{
    public:
        string adi;
};
int main(int argc, char* argv[])
{
    Ogrenci o1, *o2;
    o1.adi="Ayşe";
    o2=new Ogrenci;
    o2->adi="Ali";
    cout << o1.adi << " " << o2->adi << endl;
    return 0;
}
```

Aşırı Yükleme

```
class Islemler {
    int topla(int x, int y){
        return x+y;
    }
    int topla(int x, int y, int z){
        return x+y+z;
    }
    double topla(double x, double y){
        return x+y;
    }
};
```

Deneyin Uygulanması

```
int main(int argc, char* argv[])
{
        Complex c1(1,2);
        Complex c2(3,4);
        c1.yazdir();
        Complex c3=c1.arti(c2);
        c3.yazdir();
        Complex c4=c3.eksi(Complex(1,1));
        c4.yazdir();
        Complex c5=c4.carpi(c1);
        c5.yazdir();
        c5.eksi().yazdir();
        return 0;
}
```

Çıktı:

```
(1,2i)
(4,6i)
(3,5i)
(-7,11i)
(7,-11i)
```

Kaynaklar I


C++ overloading.
https://www.tutorialspoint.com/cplusplus/cpp_overloading.htm.