Programlama Dilleri Laboratuvar Föyü
C++ Programlama Dili 2
Nesneye Yönelik Programlama

Şevket Umut ÇAKIR

1 Giriş

C++, C dilinin özellikleri üzerine nesneye yönelik programlama öğelerini eklemiştir. Bu deneyde C++ dilinde nesneye yönelik programlama öğelerinin nasıl kullanıldığı kısaca anlatılacaktır.

2 Konu Anlatımı ve Deney Hazırlığı

C++ dilinde sınıf oluşturmak için `class` anahtar kelimesi kullanılır. Sınıf tanımlama işlemi aşağıdakine yapıda gerçekleştirilir.

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

Aşağıda örnek bir Araba sınıfının tanımlanması ve kullanılması vardır.

```
#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;
        }
};

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;
}
```

C++ üye fonksiyonları sınıfın içinde tanımlanabileceği gibi(inline) sınıfın dışında hatta başka bir dosyada bile tanımlanabilir. Aşağıda sadece yazdir fonksiyonunun sınıf dışında tanımlanmış hali ve işaretçilerle kullanımı vardır.

```
#include <iostream>
using namespace std;
class Araba
{
    public:
        string marka;
        string model;
        int modelyili;
        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;
}
```

2.1 Yıkıclar

Yıkıcılar tahsis işlemi kaldırılırken çalıştırılacak olan fonksiyonlardır. Eğer bir nesne kapsam sonuna ulaşmışsa bu nesneye tahsis kaldırma uygulanır. Yıkıcı fonksiyonlar sınıf ile aynı isimde başında ∼(tilda) sembolü ile tanımlanır. Aşağıda yıkıcı fonksiyonların tanımı ve işlevi ile ilgili bir örnek bulunmaktadır.

```
#include <iostream>
using namespace std;
class Dizi {
    public:
        Dizi(int boyut) {
            this->boyut=boyut;
            pointer = new int[boyut];
            for (int i = 0; i < boyut; ++i) {
                pointer[i]=i;
            }
        }
        ~Dizi() {
            cout << "Yıkıcı çalışıyor. Boyut:" << boyut << endl;
            delete pointer;
        }
        unsigned long operator [](int i) const    {
            return pointer[i];
        }
        void yazdir() {
            for (int i = 0; i < boyut; ++i) {
                cout << pointer[i] << " ";
            }
            cout << endl;
        }
    private:
        int * pointer;
        int boyut;
};

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

```
    }
    return 0;
}
```

Bu kod çalıştırıldığında çıktısı aşağıdaki şekilde olacaktır.

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

C++ nesnelerinin elemanlarına erişirken elimizdeki değişken bir işaretçi ise -> sembolü, aksi takdirde . sembolü kullanılır. new anahtar kelimesi dinamik bellek tahsisi için kullanılır ve geri dönüş değeri bir adrestir, bu nedenle işaretçiler ile birlikte kullanılır. Bununla ilgili aşağıda bir örnek bulunmaktadır.

```
#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;
}
```

3 Deneyin Uygulanması

Aşağıdaki kod bloğu çalıştığında beklenen çıktıyı verecek olan C++ Complex sınıfını yazınız.

```
int main(int argc, char* argv[])
{
        Complex c1(4,5);
        Complex c2(1,2);
        Complex c3=c1-c2;
        cout << c3 << endl;
        cout << c1+c2*c3 << endl;
        cout << -c2 << endl;
        cout << c1[0] << "," << c1[1] << "i" << endl;
        return 0;
}
```

Beklenen çıktı:

```
(3,3i)
(13,14i)
(-1,-2i)
4,5i
```