C++ Dersi:
Nesne Tabanlı Programlama

2. Baskı

Bölüm 15: Şablonlar

Çiğdem Turhan

Fatma Cemile Serçe

İçerik

Hedefler

- Şablon kavramının kullanım alanını açıklama
- Verilen fonksiyonun farklı veri tipindeki parametrelerle çalışması için fonksiyon şablonunu yazma
- Verilen fonksiyon şablonlarına yükleme yapma
- Sınıf şablonu tanımlama
- Sınıf şablon özelleştirme tanımı yapma
- Verilen fonksiyon tanımlamalarında hangi fonksiyon şablonunun kullanılacağını gösterme

Şablonlar

- *İng. Templates*
- *Şablonlar* aynı kod parçasını, farklı veri tipleri ile kullanılabilmemizi sağlayan bir yöntemdir.
  - fonksiyon şablonları
  - sınıf şablonları

15.1 Fonksiyon Şablonu

- *İng. Function template*
- Aynı fonksiyonun farklı tipteki parametrelerle çalışmasını *fonksiyon şablonu* kullanarak sağlayabiliriz.

```
template <class ŞablonParametresi>
ŞablonParametresi FonksiyonİsmiFonksiyonİsmi (Parametre Listesi)
{ ... }
```

- Şablonlarda parametrelerin tiplerini belirtmeyip yerine şablon parametresini veri tipi olarak kullanırız

Şablon tanımlarında `class` anahtar kelimesi yerine `typename` anahtar kelimesi de kullanılabilir.

```
template <class ŞablonParametresi>
template <typename ŞablonParametresi>
```

15.1 Fonksiyon Şablonu...

Örnek 15.1

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

template <class T>
T topla(T deger1, T deger2)      // Fonksiyon şablonu
{
    return deger1+deger2;
}
int main()
{   int x=2,y=3;
    float a=2.1,b=3.5;
    cout<<"int tipiyle cagirma: "<< topla(x,y)<<endl;
    cout<<"float tipiyle cagirma: "<< topla(a,b)<<endl;
    return 0;
}
```

Şablon parametresi

Çıktı

```
int tipiyle cagirma: 5
float tipiyle cagirma: 5.6
```

15.1 Fonksiyon Şablonu...

- **Fonksiyon Şablonlarına Yükleme** de yapabiliriz.

Örnek 15.2

```
#include <iostream>
using namespace std;
template <class A>
A enBuyuk(A bir, A iki)              // Fonksiyon şablonu yükleme-2 parametreli
{
    if (bir>iki)
        return bir;
    else
        return iki;
}
template <class A>
A enBuyuk(A bir, A iki, A uc)        // Fonksiyon şablonu yükleme-3 parametreli
{
    return enBuyuk(enBuyuk(bir,iki),uc);
}
int main()
{
    cout<<"1.cagirim: "<<enBuyuk(5.4,3.1)<<endl;      // 1. fonksiyon çağırılır
    cout<<"2.cagirim: "<<enBuyuk(10,30,20)<<endl;     // 2. fonksiyon çağırılır
    cout<<"3.cagirim: "<<enBuyuk("veli","ali","ayse")<<endl;
                                                      // 2. fonksiyon çağırılır
    return 0;
}
```

Çıktı

```
1. cagirim: 5.4
2. cagirim: 30
3. cagirim: veli
```

15.2 Sınıf Şablonu

- *İng. Class templates*
- Sınıfa ait veri ve fonksiyon üyelerinin farklı veri tipleri ile kullanımını sağlar.
- Bir sınıf şablonu herhangi bir sınıf tanımı gibi yapılır.
- Ancak sınıf başlığından hemen önce, fonksiyon şablonlarında olduğu gibi şablon ve şablon parametreleri ile tanımlanır.

```
template <class ŞablonParametresi>
    class SınıfİsmI{...}
```

15.2 Sınıf Şablonu...

Örnek 15.3

```
#include <iostream>
using namespace std;
template <class T>
class Islem                                          // Sınıf şablonu
{
    T x, y;
    public:
       Islem(T _x, T _y):x(_x),y(_y){}  // Yapıcı fonksiyon
       T kucukSayi();          // Üye fonksiyon prototipi
};
template <class T>
T Islem<T>::kucukSayi()     // Parametrelerin küçüğünü döndüren üye fonksiyon
{
    if (x < y){
        return x;
    }
    return y;
}
```

```
int main ()
{
    Islem <int> islem1(100, 75);
    cout << islem1.kucukSayi()<<endl;
    Islem <double> islem2(3.24, 8.1);
    cout << islem2.kucukSayi()<<endl;
    return 0;
}
```

Çıktı

```
75
3.24
```

15.3 Şablon Parametreleri

- Şablon parametreleri bir veya daha fazla olabilir. Birden fazla şablon parametresi “,” (virgül) işareti ile aşağıda gösterildiği gibi tanımlanır.

```
template <class T1,class T2, ... >
```

15.3 Şablon Parametreleri...

Örnek 15.4

```
#include <iostream>
using namespace std;
template <class T1, class T2>
class Islem          // İki parametreli sınıf şablonu
{
    T1 x;
    T2 y;
    public:
        Islem(T1 _x, T2 _y):x(_x),y(_y){}      // Yapıcı Fonksiyon
        T2 topla();                            // Üye fonksiyon prototipi
};
template <class T1, class T2>
T2 Islem<T1,T2>::topla()          // x ve y üyelerinin toplamını döndüren üye fonksiyon
{
    return  x + y;
}
int main ()
{
    Islem <int, double> islem(100, 3.4);
    cout << islem.topla();
    return 0;
}
```

Çıktı

```
103.4
```

15.3 Şablon Parametreleri...

Örnek 15.5

```
#include <iostream>
using namespace std;
template <class T, int N>          // Veri tipi ve değer şablon parametreleri
class Islem                        // Sınıf şablonu
{
    T dizi[N];                     // N elemanlı T tipinde dizi tanımı
    public:
        void elemanEkle(int i, T eleman);
         T elemanAl(int i);
};
template <class T, int N>
void Islem<T,N>::elemanEkle(int i, T eleman)  // dizinin i. elemanına atama yapılır
{
    dizi[i] = eleman;
}
template <class T, int N>
T Islem<T,N>::elemanAl(int i)
{
    return dizi[i];
}
```

```
int main ()
{
    Islem <int,5> islem1;
    Islem <double,10> islem2;
    islem1.elemanEkle (0,100);
    islem2.elemanEkle (3,6.9);
    cout<<islem1.elemanAl(0)<<endl;
    cout<<islem2.elemanAl(3)<<endl;
    return 0;
}
```

Çıktı

```
100
6.9
```