BTEP243 – Ders 2

FONKSİYON ŞABLONLARI

Fonksiyon şablonu, farklı veri türleri ile çalışabilen "**genel**" fonksiyondur. Programcı, fonksiyonun teknik özelliklerini tanımlarken, veri tiplerinin yerine temsili türü kullanır.

Derleyici, fonksiyon çağrıldığında, gerekli olan veri tiplerini (türlerini) işlemek için kod üretir.

Aşağıdaki aşırı yüklenmiş fonksiyonlara bir gözatalım:

```
int square (int number)
{
        return number*number;
}

double square (double number)
{
        return number*number;

}
```

Bu iki fonksiyon arasındaki tek fark, geri dönüş değerlerinin veri tipleri ve parametreleridir. Eğer şablon fonksiyonu yazarsak bu durum için daha uygun olacaktır. Şablon fonksiyonları, kullanılan her veri türü için ayrı bir fonksiyon yazma zorunluluğunu ortadan kaldırır, ve birçok farklı veri türüyle çalışan tek bir fonksiyon tanımı yazmamıza olanak tanır.

Bir şablon fonksiyonu yazarken, parametreler, geri dönüş değeri veya yerel değişkenler için gerçek tipler belirtmeniz gerekmez. Program her veri tipi için fonksiyonu çağırdığında, derleyici her veri tipi için arkaplanda ayrı fonskyionlar üretir.

Şablon Fonksiyon Yazım Kuralı:

```
template <class GenelVeriTipi>

Geri_döndürme_veri_türü FonksiyonAdi (arguman listesi)
{       // fonksyion içeriği;    }
```

Örnek1:

```
template <class T>
T square (T number)
{
  return number*number;
}

main( )
{
  int y, x=4;
  y=square(x);

  double y, d=6.2;
  y=square(d);

}
```

```
int square(int number)
{
        return number*number;
}
```

```
double square(double number)
{
        return number*number;
}
```

Örnek2:
Bu örnekte, program, her veri türü için değerleri değiştiren 3 adet swap () fonksiyonuna ve değerleri ekrana yazdıran 3 print () fonksiyonuna sahiptir. Tüm swap () işlevlerinin mantığı aynıdır.

```
#include <iostream>
using namespace std;
void swap(int& a, int& b)
{
   int temp;
   temp=a;
   a=b;
   b=temp;
}
void swap(float& a, float& b)
{
   float temp;
   temp=a;
   a=b;
   b=temp;
}
void swap(char& a, char& b)
{

   char temp;
   temp=a;
   a=b;
   b=temp;
}
```

```
void print(int a, int b)
{
   cout<<a<<"\t"<<b<<endl;
}

void print(float a, float b)
{
   cout<<a<<"\t"<<b<<endl;
}

void print(char a, char b)
{
   cout<<a<<"\t"<<b<<endl;
}
```

```
void main()
{    int x=5, y=10;
   float p=1.45, q=3.56; char ch1='a',
   ch2='b'; cout<<"before swap:\n";
   print(x,y);

   print(p,q);
   print(ch1,ch2);

   swap(x,y); swap(p,q);
   swap(ch1,ch2); cout<<"after
   swap\n";

   print(x,y); print(p,q); print(ch1,ch2);
}
```

Yukarıdaki fonksiyonlar için Şablon Fonksiyonu kullanalım;

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

template<class T>
void swap(T& a, T& b)
{   T    temp;    temp=a;
   a=b;
   b=temp;
}
template<class W>
 void print(W a, W b)
 {    cout<<a<<"\t"<<b<<endl;     }

 void main()
 {    int x=5, y=10;
    float p=1.45, q=3.56; char
    ch1='a', ch2='b';
    cout<<"before swap:\n";
    print(x,y);
    print(p,q);
    print(ch1,ch2);
    swap(x,y); swap(p,q);
    swap(ch1,ch2);
    cout<<"after swap\n";
 print(x,y); print(p,q); print(ch1,ch2);
 }
```

Örnek3:

Üç farklı türde sayının toplamını bulan ve sonucu ekrana yazdıran bir şablon fonskiyon yazınız.

```
#include<iostream>
template<class T, class P ,class Q>

void sum(T num1, P num2, Q num3)
{ cout<<num1+num2+num3<<"\n"; }

void main()
{    int a=3,b=4,c=5;
     float x=2.3, y=3.7,z=5.8;
     sum(a,b,c);  //call sum(), sended int,int,int values
     sum(a,x,y); //send int,float,float values
     sum(x,b,y); //float,int,float values
     sum(x,y,z); //float float float values
     sum(x,z,c); float,float,int values
}
```

sum() fonksiyonu çağrıldığında, derleyici aşağıdaki fonksiyonları oluşturur:

```
sum(int num1, int num2, int num3)
{ cout<<num1+num2+num3<<"\n"; }

sum(int num1, float num2, float num3)
{ cout<<num1+num2+num3<<"\n"; }

sum(float num1, int num2, float num3)
{ cout<<num1+num2+num3<<"\n"; }

sum(float num1, float num2, float num3)
{ cout<<num1+num2+num3<<"\n"; }

sum(float num1, float num2, float num3)
{ cout<<num1+num2+num3<<"\n"; }
```

Örnek4

Bir dizideki (array) en küçük sayıyı bulan bir şablon fonksiyon yazınız.

```
template <class T>
T minimum(T arr[ ], int size)
{
        T smallest=arr[0];
        for( int k=1; k<size; k++)
        {
                if(arr[k]<smallest)
                smallest=arr[k];
        }
        return smallest;
```

```
}
void main( )
{
        int arr1[ ]={40,20,35};
        double arr2[ ]={40.25,28.13,18.21,50.10};
        cout<<"The minimum value of arr1:"<<minimum(arr1,3);
        cout<<"The minimum value of arr2:"<<minimum(arr2,4);
}
```

Örnek5:

```
template <class T>
T sum(T val1, T val2)
{
        return val1+val2;
}
template <class T>
T sum(T val1, T val2, T val3)
{
        return val1+val2+val3;
}

void main( )
{
        double num1, num2, num3;
        cout<<"Enter 2 values:";
        cin>>num1>>num2;
        cout<<"Sum:"<<sum(num1,num2);
        cout<<"Enter 3 values:";
        cin>>num1>>num2>>num3;
        cout<<"Sum:"<<sum(num1,num2,num3);
}
```

Örnek6:

```
// function template
#include <iostream>
using namespace std;
template <class T>

T GetMax (T a, T b)

{ T result;
result = (a>b)? a : b;
return (result);
}

int main () {
int i=5, j=6, k;
```

```
long l=10, m=5, n;
k=GetMax<int>(i,j);
n=GetMax<long>(l,m);
cout << k << endl;
cout << n << endl;
return 0; }
```