

MALATYA
TURGUT ÖZAL ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
ALGORİTMA VE PROGRAMLAMA 2

BAŞVURULAR

BAŞVURULAR

- Şuana kadar fonksiyonlarda değer ile çağırma ve referans ile çağırma yöntemlerini görmüştük.
- Değer ile çağırma yönteminde, fonksiyon çağrılırken kullanılan parametrelerin kopyaları fonksiyona gönderiliyordu.
- Referans ile çağırma yönteminde ise fonksiyonları parametreleri göstericiler olacak şekilde tanımlıyorduk ve çağırmak için parametre olarak değişkenlerin adreslerini gönderiyorduk.
- C++ da bunların dışında bir çağırma yöntemi daha vardır.
- Bu yöntemin adı başvurudur.

BAŞVURULAR

- Aslında basit olarak düşündüğümüzde başvurular göstericiler ile aynı işi yaparlar.
- Fakat göstericilere göre bazı üstünlükleri vardır.
- Anlaşılır programlar yazabilmek için daha uygundurlar.
- Şimdi göstericiler ile başvuruları bir örnek üzerinde görelim.

Örnek 1

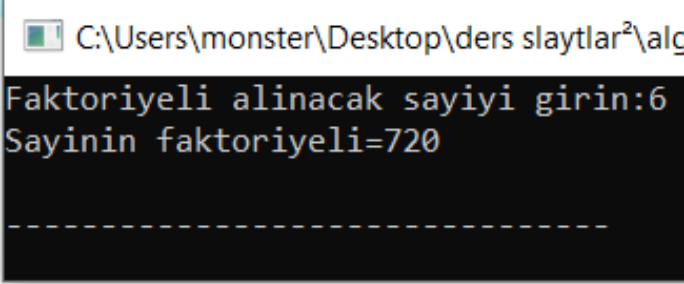
```
1  #include <iostream>
2  using namespace std;
3
4  void gosterici_fonk(double *);
5  void basvuru_fonk(double &);
6
7  int main() {
8
9      double x=1;
10
11      gosterici_fonk(&x);
12      cout<<"Gosterici ile cagirdiktan sonra x' in degeri = "<<x<<endl;
13
14      basvuru_fonk(x);
15      cout<<"Basvuru ile cagirdiktan sonra x' in degeri = "<<x<<endl;
16
17      return 0;
18 }
19
20 void gosterici_fonk(double *a){
21     cout<<"Gosterici fonk icindeyiz....."<<endl;
22     *a=5;
23 }
24
25 void basvuru_fonk(double &b){
26     cout<<"Basvuru fonk icindeyiz....."<<endl;
27     b=25;
28 }
```

```
C:\Users\monster\Desktop\ders slaytlar2\algoritma2\ders slayt\kodl
Gosterici fonk icindeyiz.....
Gosterici ile cagirdiktan sonra x' in degeri = 5
Basvuru fonk icindeyiz.....
Basvuru ile cagirdiktan sonra x' in degeri = 25
```

- Öncelikle double türeden bir x değişkenine ilkdeğer olarak 1 atanıyor ve sırasıyla önce gösterici_fonk(), sonrada basvuru_fonk() çağırılıyor. Fakat ikisininde çağırılma yöntemleri biraz farklı.
- İlk çağrılan fonksiyon olan gösterici_fonk() parametre olarak double bir değişkene işaret eden gösterici(pointer) alan ve geriye geriye değer döndürmeyen bir fonksiyon olarak tanımladık.
- Böyle bir fonksiyonu çağırmak için fonksiyona parametre olarak bir gösterici göndermeliyiz.
- Bu fonksiyonun içerisinde bir değişkene değer atamak için ise değişken adından önce * koymak zorundayız.
- Çünkü fonksiyona parametre olarak gelen bir adrestir ve * operatörü bu adresteki veriyi işaret eder.
- İkinci fonksiyon olan basvuru_fonk() da durum biraz farklıdır.
- Başvuru işleminde x değişkeni önüne & konulmadı. Yani değişken direkt olarak fonksiyonu çağırmada parametre olarak kullanıldı.
- Fonksiyon içerisinde bu değişkene atama yaparken * kullanmayada gerek kalmadı, direkt olarak değişkene atama yapıldı.
- Bu & ve * operatörlerinde kurtulmamızın nedeni, bu operatörlerin yaptığı işleri başvurular sayesinde derleyicinin bizim için yapmış olmasıdır.

Örnek 2: Faktöriyel hesabı yapan bir fonksiyonu başvurular ile oluşturalım.

```
1  #include <iostream>
2  using namespace std;
3
4  void faktoriyel(int &);
5
6  int main() {
7
8      int x;
9
10     cout<<"Faktoriyeli alınacak sayiyi girin:";
11     cin>>x;
12
13     faktoriyel(x);
14     cout<<"Sayinin faktoriyeli="<<x<<endl;
15
16     return 0;
17 }
18
19 void faktoriyel(int &b){
20     int fk=1;
21     int i;
22
23     for (i=1;i<=b;i++){
24         fk=fk*i;
25     }
26
27     b=fk;
28 }
```



C:\Users\monster\Desktop\ders slaytlar²\alg
Faktoriyeli alınacak sayiyi girin:6
Sayinin faktoriyeli=720

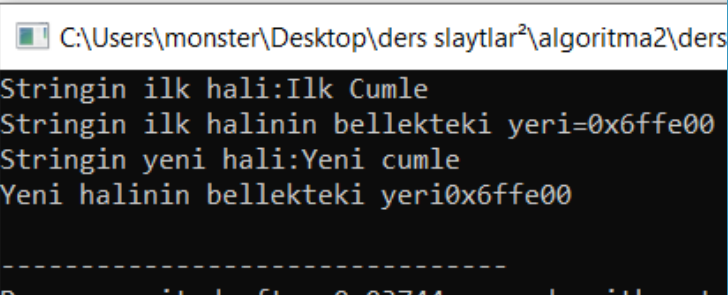
- İlk örneğimizde olduğu gibi fonksiyonumuza direkt olarak bir değişken gönderdik.
- Fakat derleyici onu bir adres olarak bildirdi.
- Fonksiyon içerisinde fk değişkeni ile faktöriyel hesapladık ve sonra fonksiyona mainden gelen değişkene o fk değerini atadık.
- Son olarak main içinde aynı değişkeni ekrana yazdırdığımızda faktöriyel alınmış halini ekranda görmüş olduk.

FONKSİYONLARI VERİLERİ BAŞVURU OLARAK DÖNDÜRMESİ

- Başvurular, fonksiyonlara parametre olarak gönderildiği gibi fonksiyonların geri dönüş değeri de olabilirler.
- Geri dönüş değeri olarak başvuru döndüren fonksiyonlarda aktarım adresler ile yapılır,
- Fakat bu geri dönüş değeri normal değişken gibi kullanılabilir.

ÖRNEK 3:

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 string& degistir(string &);
6
7 int main() {
8     string a="Ilk Cumle";
9
10    cout<<"Stringin ilk hali:"<<a<<endl;
11    cout<<"Stringin ilk halinin bellekteki yeri="<<&a<<endl;
12
13    cout<<"Stringin yeni hali:";
14    cout<<degistir(a)<<endl;
15
16    cout<<"Yeni halinin bellekteki yeri"<<&degistir(a)<<endl;
17
18
19    return 0;
20 }
21
22 string& degistir(string &cumle){
23     cumle="Yeni cumle";
24     return cumle;
25 }
```



- Bu örneğimizde ilk olarak bir string olan a değişkeninin oluşturduk ve ilk değer olarak «ilk cümle» yazdık.
- a stringini degistir() fonksiyonuna parametre olarak gönderdik.
- Degistir() başvuru ile parametre alan ve geriye başvuru döndüren bir fonksiyondur.
- Fonksiyon parametre olarak aldığı a stringinin adresinin içeriğine «yeni cümle» yazdı ve geriye başvuru olarak yine bu adresi döndürdü.
- Böylece main() içinde cout<<degistir(a) şeklinde bu adresteki veriyi ekran yazdırdık.
- Bu konuyu daha iyi anlayabilmemiz için fonksiyona gönderilen stringin adresini ve geri dönüş adresini ekran yazdırdık.
- Ekran çıktısında görüldüğü gibi ikisinin de adresi aynı olduğunu gördük.

BAŞVURULARIN NESNELERLE KULLANIMI

- Başvurular değişkenler gibi nesneler ile kullanılabilir.
- Önceki haftalarda sınıflar ve nesneler konusunda anlattığımız gibi fonksiyonlara parametre olarak nesne geçirme işlemlerini genellikle değer çağırma yöntemi ile yaptık.
- Bir fonksiyonu, başvurular ile çağrılacak şekilde tasarlırsak derleyici, fonksiyona nesnenin adresini gönderir.
- Böylece fonksiyon için orijinal değişkenin kopyası oluşturulmaz.

ÖRNEK 4-BAŞVURULARIN NESNELERLE KULLANIMI

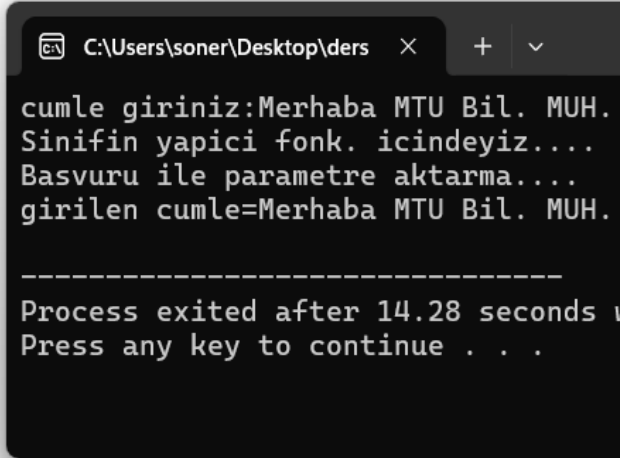
Bu örneğimizde başvuru ile fonksiyona nesne aktarımını gördük.

1- Main içinde ilk olarak bir string cümle değişkeni oluşturduk. Stringlerde tamamıyla bir satırdaki cümleleri alma olmadığı için getline fonksiyonunu kullandık. Bu fonksiyon getline(cin,cümle) şeklinde kullanılır ve klavyeden girilen her girişi cümleye kopyalar.

2- cagir() sınıfının yapıcı fonksiyonuna girilen stringi parametre olarak gönderdik. Yapıcı fonksiyon parametre olarak aldığı bu stringi sınıfın private olan a değişkenine atadı ve ekrana yapıcı fonksiyonun çalıştığını bildirdi.

3- basvuru_fonk() fonksiyonu parametresi x nesnesi olacak şekilde başvuru ile çağrıldı. Derleyici fonksiyona orijinal nesnemiz olan x in adresini gönderir. Eğer fonksiyonu başvuru ile kullanmasaydık yeni bir kopya nesne oluşturulacaktı.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class cagir{
6      string a;
7
8      public:
9          cagir(string x){
10             a=x;
11             cout<<"Sinifin yapici fonk. icindeyiz..."<<endl;
12         }
13
14         string erisim(){
15             return a;
16         }
17     };
18
19     void basvuru_fonk(cagir &);
20
21     int main() {
22         string cumle;
23         cout<<"cumle giriniz:";
24         getline(cin,cumle);
25
26         cagir x(cumle);
27
28         basvuru_fonk(x);
29
30         return 0;
31     }
32
33     void basvuru_fonk(cagir &nesne){
34         cout<<"Basvuru ile parametre aktarma..."<<endl;
35         cout<<"girilen cumle="<<nesne.erisim()<<endl;
36     }
37
```



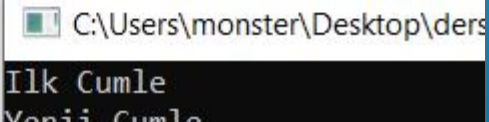
C:\Users\soner\Desktop\ders x + v

cumle giriniz:Merhaba MTU Bil. MUH.
Sinifin yapici fonk. icindeyiz....
Basvuru ile parametre aktarma....
girilen cumle=Merhaba MTU Bil. MUH.

Process exited after 14.28 seconds with return code 0
Press any key to continue . . .

ÖRNEK 5-BAŞVURULARIN NESNELERLE KULLANIMI

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class test{
6      string a;
7
8      public:
9          test(string x){
10             cout<<x<<endl;
11             a=x;
12         }
13
14         string& degistir(test &y){
15             y.a="Yenii Cumle";
16             return y.a;
17         }
18     };
19
20 int main() {
21     test nesne1("Ilk Cumle");
22
23     cout<<nesne1.degistir(nesne1)<<endl;
24
25     return 0;
26 }
27
28
```



- Tanımlanan test sınıfında önce yapıcı fonksiyon ile private değişken olan a ya atama yaptık.
- Daha sonra degistir() fonksiyonu, daha önce oluşturulan nesne1, fonksiyonun parametresi olacak şekilde başvuru ile çağrıldı.
- Fonksiyon içerisinde başvuru ile gelen nesneye atama yapıldı ve geri dönüş değeri olarak başvuru ile gelen nesnenin private değişkeni döndürüldü ve ekranda yazdırıldı.